Methods and Tools for Weak Problems of Translation

Muhammad Ghulam Abbas Malik

▶ **To cite this version:**

Muhammad Ghulam Abbas Malik. Methods and Tools for Weak Problems of Translation. Computer Science [cs]. Université Joseph-Fourier - Grenoble I, 2010. English. NNT : . tel-00502192

HAL Id: tel-00502192
https://theses.hal.science/tel-00502192

Submitted on 13 Jul 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université de Grenoble

N° attribué par la bibliothèque

/_ /_ /_ /_ /_ /_ /_ /_ /_ /

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité: "INFORMATIQUE" (Computer Science)

dans le cadre de

l'École Doctorale "Mathématiques, Sciences et Technologie de l'Information, Informatique"

(Doctoral School "Mathematics, Information Science and Technology, Computer Science")

présentée et soutenue publiquement

par

Muhammad Ghulam Abbas MALIK

le 9 juillet 2010

Méthodes et outils pour les problèmes faibles de traduction

(Methods and Tools for Weak Problems of Translation)

Thèse dirigée par

M. Christian BOITET	**Directeur de thèse**
M. Pushpak BHATTACHARYYA	**Codirecteur de thèse**

JURY

M. Arthur SOUCEMARIANADIN	Président
Mme Violaine PRINCE	Rapporteur
M. Patrice POGNAN	Rapporteur
M. Eric WEHRLI	Rapporteur
M. Vincent BERMENT	Examinateur
M. Alain DÉSOULIÈRES	Examinateur
M. Christian BOITET	Directeur
M. Pushpak BHATTACHARYYA	Codirecteur

Préparée au laboratoire GETALP–LIG (CNRS–INPG–UJF–UPMF)

Table of Contents

Introduction **1**

Part I **7**

Scriptural Translation and other Weak Translation Problems **7**

Chapter 1. Scriptural Translation **9**

1.1. Scriptural Translation, Transliteration and Transcription 9
1.2. Machine Transliteration in Natural Language Processing (NLP) 9
1.2.1. Grapheme-based models 10
1.2.2. Phoneme-based models 11
1.2.3. Hybrid and correspondence-based models 11
1.3. Scriptural Translation 12
1.3.1. Challenges and barriers within the same language 14
1.3.1.1. Scriptural divide 14
1.3.1.2. Under-resourced and under-written languages 15
1.3.1.3. Absence of necessary information 15
1.3.1.4. Different spelling conventions 16
1.3.1.5. Transliterational or transcriptional ambiguities 17
1.3.2. Challenges and barriers within dialects 17
1.3.2.1. Distinctive sound inventories 17
1.3.2.2. Lexical divergence and translational ambiguities 18
1.3.3. Challenges and barriers between closely related languages 19
1.3.3.1. Characteristics 19
1.3.3.2. Under-resourced dialect or language pairs 19
1.4. Approaches for Scriptural Translation 20
1.4.1. Direct programming approaches 20
1.4.2. Finite-state approaches 21
1.4.3. Empirical, machine learning and SMT approaches 21
1.4.4. Hybrid approaches 21
1.4.5. Interactive approaches 22

Chapter 2. Scriptural Translation Using FSTs and a Pivot UIT **23**

2.1. Scriptural Translation for Indo-Pak Languages 24
2.1.1. Scripts of Indo-Pak languages 24
2.1.1.1. Scripts based on the Persio-Arabic script 24
2.1.1.2. Indic scripts 25
2.1.2. Analysis of Indo-Pak scripts for scriptural translation 26
2.1.2.1. Vowel analysis 26
2.1.2.2. Consonant analysis 27
2.1.2.2.1. Aspirated consonants 27
2.1.2.2.2. Non-aspirated consonants 28
2.1.2.3. Diacritics analysis 28
2.1.2.4. Punctuations, digits and special characters analysis 29
2.1.3. Indo-Pak scriptural translation problems 29
2.2. Universal Intermediate Transcription (UIT) 29
2.2.1. What is UIT? 30
2.2.2. Principles of UIT 31
2.2.3. UIT mappings for Indo-Pak languages 31
2.3. Finite-state Scriptural Translation model 32

2.3.1. Finite-state transducers 32
2.3.1.1. Non-deterministic transducers 33
2.3.1.1.1. Character mappings 33
2.3.1.1.2. Contextual mappings 37
2.3.1.2. Deterministic transducers 37
2.3.2. Finite-state model architecture 38
2.4. Experiments and results 39
2.4.1. Test data 39
2.4.2. Results and discussion 41
2.5. Conclusion 47

Part II 49

Statistical, Hybrid and Interactive Models for Scriptural Translation 49

Chapter 3. Empirical and Hybrid Methods for Scriptural Translation 51

3.1. SMT Model for Scriptural Translation 51
3.1.1. Training data 52
3.1.2. Data alignments 53
3.1.2.1.1. Character alignments 53
3.1.2.1.2. Cluster alignments 54
3.1.3. Translation models 57
3.1.4. Target language models 58
3.1.4.1. Word language models 58
3.1.4.2. Sentence language models 58
3.1.5. SMT systems for Hindi-Urdu scriptural translation 59
3.1.6. Experiments and results 60
3.1.6.1. Experiments 61
3.1.6.2. Results and discussion 61
3.1.6.2.1. Hindi to Urdu results 61
3.1.6.2.2. Urdu to Hindi results 65
3.1.6.3. Effects of different parameters on SMT results 67
3.2. Hybrid model for scriptural translation 69
3.2.1. Monolingual data 69
3.2.2. Language models 69
3.2.3. Word ambiguity network 70
3.2.4. Hybrid Model for scriptural translation 72
3.2.4.1. Finite-state scriptural translation 72
3.2.4.1.1. Statistical disambiguator 72
3.2.5. Experiments and results 73
3.3. Comparisons of different models 73
3.3.1. Finite-state vs. SMT 74
3.3.2. Finite-state vs. Hybrid 75
3.3.3. SMT vs. Hybrid 75
3.4. Conclusion 76

Chapter 4. Interactive Scriptural Translation 77

4.1. Interactive Machine Translation 77
4.2. Motivation 78
4.3. Interactive Finite-state Scriptural Translation System 78
4.3.1. System Architecture 79
4.3.2. Weighted finite-state scriptural translation 80
4.3.2.1. Weighted finite-state transducers 81
4.3.2.2. Word-to-word scriptural translation transducers 82

4.3.3. Interactive GUI 84
4.4. Evaluation Design for Interactive Scriptural Translation Model 86

Part III 87

Interdialectal Translation–a Weak Translation Problem 87

Chapter 5. Interdialectal Translation –another Weak Translation Problem 89

5.1. Problems of Interdialectal Translation 89
5.1.1. Lexical divergence and word-to-word translation 89
5.1.2. Under-resourcedness 90
5.2. Analysis of Interdialectal Translation 90
5.2.1. Word-to-word translation decision 91
5.2.2. Agreement recomputation 91
5.2.2.1. Short distance agreements 92
5.2.2.2. Long distance agreements 92
5.2.3. Partial or complete syntax analysis 93
5.3. Computational Models for Interdialectal Translation 93
5.3.1. Word-to-word translation 93
5.3.2. Partial or complete syntactic analysis 94
5.3.3. Tree transduction and agreement recomputation 95
5.3.4. Syntactic translation model for interdialectal translation 96
5.4. Scope of Interdialectal Translation 97

Chapter 6. Towards an Interdialectal Translation System 99

6.1. Approaches for Building an IDTS 99
6.1.1. Empirical approach 99
6.1.2. Syntactic translation approach 100
6.2. Building Parallel Resources for Interdialectal Translation on Web 101
6.2.1. Parallel lexicon 101
6.2.2. Parallel corpus 102
6.3. Building Linguistic Components 103
6.3.1. Morphological analyzer 103
6.3.2. Syntactic and Syntax tree transformation grammars 103
6.4. Conclusion 104

Conclusion and Perspectives 105

Bibliography 107

Annexes 119

Annex 1 Analysis of Writing Systems: Urdu, Punjabi/Shahmukhi, Seraiki/Shahmukhi and Kashmiri 121

Analysis of Urdu 121
Brief History of Urdu 121
Urdu Alphabet 122
Urdu Alphabet Standard 122
Analysis of Urdu script 124
Consonants 124
Non-aspirated consonants 124
Aspirate consonants 124
Vowels 125
Diacritical Marks 125

HAMZA 126
Analysis of Punjabi/Shahmukhi 126
Analysis of Seraiki/Shahmukhi 127
Analysis of Kashmiri/Urdu 127
Analysis of Sindhi/Sindhi 127

Annex 2 Analysis of the Hindi Writing Systems 129

Hindi Alphabet 129
Historical emergence of the Hindi/Devanagari script 129
Features of Devanagari script 129
Standard codepage of Hindi 130
Analysis of Devanagari for Hindi 130
Vowels 131
Vowel diacritical marks 131
Independent Vowels 132
Consonants 133
Non-aspirated consonants 133
Aspirated consonants 134
Conjunct forms 134

Annex 3 Analysis of Punjabi/Gurmukhi 137

Annex 4 Analysis of Devanagari for Sindhi, Seraiki and Kashmiri 139

Annex 5 UIT Mappings for Indo-Pak Languages 141

Urdu UIT Mappings 141
Punjabi UIT Mappings 141
Consonant mappings 141
Vowel mappings 142
Other symbols 143
Seraiki UIT Mappings 144
Sindhi UIT Mappings 144
Hindi UIT Mappings 144
Consonant mappings 144
Vowel mappings 145

Annex 6 Finite-state Transducers 147

Hindi to UIT Finite-state Transducer 147
UIT to Hindi Finite-state Transducer 149
Urdu to UIT Finite-state Transducer 152
UIT to Urdu Finite-state Transducer 160
Punjabi/Shahmukhi to UIT Finite-state Transducer 162
UIT to Punjabi/Shahmukhi Finite-state Transducer 170
Punjabi/Gurmukhi to UIT Finite-state Transducer 173
UIT to Punjabi/Gurmukhi Finite-state Transducer 176
Seraiki/Shamukhi to UIT Finite-state Transducer 180
UIT to Seraiki/Shahmukhi Finite-state Transducer 187
Seraiki/Devanagari to UIT Finite-state Transducer 190
UIT to Seraiki/Devanagari Finite-state Transducer 192
Word Ambiguity Transducer 196

Annex 7 Results 199

Hindi to Urdu Results 199
Character alignment results 199
Cluster alignment results 201
BLEU and NIST scores 203

Urdu to Hindi Results 207
Character alignment results 207
Cluster alignment results 210
NIST and BLEU Scores 212

French Summary 217

Extended French Summary 233

Résumé 245

Abstract 246

List of Figures

Figure 1: Vauquois' triangle [21, 26, 28, 178, 179] 3
Figure 2: Adopted and adapted Vauquois's triangle for the weak translation problem 4
Figure 3: Four steps in English-to-Chinese transliteration of names [180] 11
Figure 4: Part of the decision tree for $\Phi_{(SP)T}$ [146] 12
Figure 5: Languages with different writing systems 13
Figure 6: Hindi - Urdu example sentence 13
Figure 7: Example of missing information from the source side 16
Figure 8: Hindi examples with short vowels at the end 16
Figure 9: Sindhi Transliteration Example 17
Figure 10: Hindi-Urdu example for dialectal translation 19
Figure 12: XFST code example 20
Figure 11: Direct programming example [125] 20
Figure 13: Family tree of the Indo-Pak and other Pakistani languages 24
Figure 14: Context sensitive shapes of Beh [194] 25
Figure 15: BEH with common diacritics 25
Figure 16: Hindi, Urdu and Sindhi vowel chart (source Wikipedia) 26
Figure 17: Ambiguity due to missing diacritical marks 29
Figure 18: Example of Punjabi/Shahmukhi - Punjabi/Gurmukhi pair 29
Figure 19: Ambiguities of Hindi/Devanagari to Punjabi/Shahmukhi scriptural translation 33
Figure 20. FST for XFST code of Table 18 36
Figure 21: Contextual replace operator of XFST 37
Figure 22: System architecture of the finite-state model for Indo-Pak scriptural translation 38
Figure 23: Unique output of the sample run by determintisc FSTs 39
Figure 24: Correct scriptural translation of the sample run 39
Figure 25: A part of confusion network for the ambiguous outputs of the non-deterministic FSTs 39
Figure 26: A sample Hindi source text for Hindi to Urdu scriptural translation 41
Figure 27: Classification scale based on the word accuracy for scriptural transaltion 43
Figure 28: Classification scale based on the sentence accucary rate for scriptural translation 43
Figure 29: Sample Entries from [151] 52
Figure 30. SMT Systems for Hindi to Urdu scriptural translation 59
Figure 31. SMT systems for Urdu to Hindi scriptural translation 60
Figure 32. Effect of cluster alignments on SMT systems from Hindi to Urdu scriptural translation 67
Figure 33. Effect of cluster alignments on SMT systems from Urdu to Hindi scriptural translation 67

Figure 34. Effect of post-procsseding on Hindi ot Urdu SMT systems with cluster alignments 68
Figure 35. Effect of reordering and tunning on Urdu to Hindi scriptural translation 68
Figure 36. Word confusion network example 70
Figure 37. Example rules for generating a word confusion network 70
Figure 38. Possible word variations for the example word 71
Figure 39. A sample word map 71
Figure 40. Hybrid model for scriptural translation 72
Figure 41. Example of Urdu sentences with their Hindi reference for our hybrid model 72
Figure 42. UHT-FSM output 72
Figure 43. Statistical disambiguation process 73
Figure 44. Example Hindi output of the hybrid model 73
Figure 45. Comparison of SMT and FST systems for Hindi to Urdu translation 74
Figure 46. Comparison of SMT and FST systems for Urdu to Hindi translation 74
Figure 47. Comparison of FST and hybrid systems for Urdu to Hindi scriptural translation 75
Figure 48. Comparison of SMT and hybrid systems for Urdu to Hindi scriptural translation 75
Figure 49. System architecture of interactive finite-state scriptural translation system 79
Figure 50. Interactive graphical user interface 79
Figure 51. Weighted finite-state scriptural translation with the help of translation memories ... 80
Figure 52. A sample finite-state transducer for weighted finite-state scriptural transaltion 82
Figure 53. A sample weighted confusion network for the example sentence 82
Figure 54. A sample word-to-word transducer 83
Figure 55. Ambiguous nature of word-to-word scriptural translation transducers 83
Figure 56. Interactive graphical user interface of IFSTS after translation process 84
Figure 57. All possible word options in IFSTM 85
Figure 58. Scale of user satisfaction 86
Figure 59. Scale of user satisfaction 86
Figure 60. Examples of Quebecois to French word-to-word translation with gender adjustments 91
Figure 61. Example of Hindi–Urdu word-to-word translation without agreement adjustments . 91
Figure 62. Examples of Quebecois to French word-to-word translation without gender adjustments 91
Figure 63. Example of Hindi–Urdu word-to-word translation 92
Figure 64. Example of Hindi–Urdu word-to-word translation 93
Figure 65: A parsed sentence [44] 94
Figure 66: Proposed interdialectal translation model 96
Figure 67: Adapted Vauquois' triangle 100
Figure 68: Interactive scriptural translation interface after automatic translation process 101
Figure 69: Interactive scriptural translation interface for adding equivalent words 102

Figure 70: Comparison of Urdu to Hindi scriptural translation systems at word and sentence level 105
Figure 71: Urdu Zabta Takhti 123
Figure 72: An example Hindi sentence 130
Figure 73: Example sentence of Punjabi in Gurmukhi script 137

List of Tables

Table 1: Subproblems of the weak translation problem (by order of increasing complexity) 3
Table 2:Results of Urdu toHindi scriptural translation 5
Table 3: Scale based on sentence accuracy for scriptural translation quality 6
Table 4: English-Chinese direct orthographic mapping example [65] 10
Table 5: Subproblems of scriptural translation for general translation and weak translation 14
Table 6: Population of Hindi and Urdu [156] 15
Table 7: Examples from English and Arabic distinctive sound inventories 17
Table 8: Japanese Korean vowel comparison 18
Table 9: Hindi and Marathi vowel comparison 18
Table 10: Additional Kashmiri vowel characters and diacritics 26
Table 11: Vowels analysis for Urdu and Punjabi in scripts based on the Persio-Arabic script [119, 121] 27
Table 12: Numbers of aspirated consonants in Indo-Pak languages 28
Table 13: One native sound for multiple characters 28
Table 14: Example of scriptural translation between Indo-Pak languages 30
Table 15: UIT encoding of aspirated consonants of the Indo-Pak languages 32
Table 16: Regular rules for aspirated consonants of Hindi and Punjabi/Shahmukhi 34
Table 17: Regular relations for the example of Figure 19 34
Table 18: Sample XFST code for Hindi to UIT scriptural translation 35
Table 19: Sample XFST code for UIT to Punjabi/Shahmukhi scriptural translation 35
Table 20: Sample XFST code for Urdu to UIT scriptural translation 37
Table 21: Corpus word frequncies of confusing characters [121] 37
Table 22: Sample run of finite-state model for scriptural translation from Hindi to Urdu 38
Table 23: Data sets for testing scriptural translaiton 40
Table 24: Result of Hindi to Urdu scriptural translation by the finite-state system 41
Table 25: Default Urdu output of Hindi to Urdu scriptrual translation 42
Table 26: Processed Urdu output of Hindi to Urdu scriptrual translation 42
Table 27: Result of Urdu to Hindi scriptural translation by the finite-state system 44
Table 28: Hindi output for Urdu input text with diacritics 44
Table 29: Hindi output for Urdu input text without diacritics 45
Table 30: Results of Punjabi scriptural translation by the finite-state system 46
Table 31: Results of Seraiki scriptural translation by the finite-state system 46
Table 32: BLEU and NIST scores for scriptural translation of Hindi - Urdu and Punjabi 47
Table 33: Examples from a Hindi-Urdu parallel lexicon 53
Table 34: Examples of Hindi - Urdu parallel words 53

Table 35: Character alignment examples from Hindi to Urdu with diacritics 54
Table 36: Vowel alignment from Urdu with diacritics to Hindi 55
Table 37. Urdu character sequeces for vowel alignments 55
Table 38. Gemination alignment from Urdu to Hindi 55
Table 39. Aspirated consonant alignment from Urdu to Hindi 56
Table 40: Hindi - Urdu example words for alignment 56
Table 41. Cluster alignments examples from Hindi to Urdu with diacritics 57
Table 42. HU Test Set 2 results of Hindi to Urdu SMT systems (character alignments) 62
Table 43: Sample Hindi to Urdu outputs for SMT systems with the best results 63
Table 44: HU Test Set 3 best results of Hindi to Urdu SMT systems (character alignments) 64
Table 45: HU Test Set 2 and 3 best results of Hindi to Urdu SMT systems (cluster alignments) 64
Table 46: HU Test Set 2 and 3 best results of Urdu to Hindi SMT systems (character alignments) 65
Table 47: A sample SMT system Hindi output with its reference 66
Table 48: A sample SMT system Hindi output with its reference for Urdu input without diacritics 66
Table 49. No. of Ambiguites in word map 71
Table 50. Results o hybrid model for HU Test Set 2 73
Table 51. Sample run of weighted scriptural translation 81
Table 52: Frequency analysis of Urdu corpus 81
Table 53. Sample XFST code for word-to-word scriptural translation transducer 82
Table 54. Objective and subjective measures for IFSTS 86
Table 55. Subproblems of interdialectal translation 89
Table 56. Sample French - Quebecois parallel word list 90
Table 57. Sample parallel lexicon for Hindi - Urdu 91
Table 58. Short distance agreements 92
Table 59. Examples of long distance agreements 92
Table 60: A sample context-free grammarlearnt from tree bank [44] 95
Table 61: Number of speakers of Indo-Pak languages 106
Table 62: Non-aspirated Urdu consonants 124
Table 63: Aspirated Urdu consonants 125
Table 64: Analysis of diacritical marks 126
Table 65: Kashmiri characters 127
Table 66: Aspirated and non-aspirated Sindhi consonants 127
Table 67: Standard codepage of Hindi (Unicode) 130
Table 68: Analysis of devangari vowel dicritics 132
Table 69:Analysis of Devangari vowel characters 133

Table 70: Non-aspirated Hindi consonants 134
Table 71: Aspirated Hindi consonants 134
Table 72: Unicode codepage for Gurmukhi 138
Table 73: Punjabi/Shahmukhi consonants mappings 142
Table 74: Vowel mappings at the start of a word 142
Table 75: Vowel mappings in the middle or at the end of a word 143
Table 76: Vowel sign mappings 143
Table 77: Other symbol mappings 143
Table 78: Seraiki mappings for additional characters 144
Table 79: UIT mappings of additional Sindhi/Sindhi and Sindhi/Devanagari characters 144
Table 80: UIT mappings of Hindi/Devanagari consonants 145
Table 81: UIT mappings of Hindi vowel characters 145
Table 82: UIT mappings of Hindi vowel signs 145
Table 83: HU Test Set 1 results of Hindi to Urdu SMT systems (character alignments) 199
Table 84: HU Test Set 3 results of Hindi to Urdu SMT systems (character alignments) 200
Table 85: HU Test Set 1 results of Hindi to Urdu SMT systems (cluster alignments) 201
Table 86: HU Test Set 2 results of Hindi to Urdu SMT systems (cluster alignments) 202
Table 87: HU Test Set 3 results of Hindi to Urdu SMT systems (cluster alignments) 203
Table 88: HU Test Set 1 results of Hindi toUrdu SMT System (Character Alignments) 204
Table 89: HU Test Set 1 results of Hindi toUrdu SMT System (Cluster Alignments) 204
Table 90: HU Test Set 2 results of Hindi toUrdu SMT System (Character Alignments) 205
Table 91: HU Test Set 2 results of Hindi toUrdu SMT System (Cluster Alignments) 205
Table 92: HU Test Set 3 results of Hindi toUrdu SMT System (Character Alignments) 206
Table 93: HU Test Set 3 results of Hindi toUrdu SMT System (Cluster Alignments) 206
Table 94: HU Test Set 1 results of Urdu to Hindi SMT systems (character alignments) 207
Table 95: HU Test Set 2 results of Urdu to Hindi SMT systems (character alignments) 208
Table 96: HU Test Set 3 results of Urdu to Hindi SMT systems (character alignments) 209
Table 97: HU Test Set 1 results of Urdu to Hindi SMT systems (cluster alignments) 210
Table 98: HU Test Set 2 results of Urdu to Hindi SMT systems (cluster alignments) 211
Table 99: HU Test Set 3 results of Urdu to Hindi SMT systems (cluster alignments) 212
Table 100: HU Test Set 1 results of Urdu to Hindi SMT systems (character alignments) 213
Table 101: HU Test Set 1 results of Urdu to Hindi SMT systems (cluster alignments) 213
Table 102: HU Test Set 2 results of Urdu to Hindi SMT systems (character alignments) 214
Table 103: HU Test Set 2 results of Urdu to Hindi SMT systems (cluster alignments) 214
Table 104: HU Test Set 3 results of Urdu to Hindi SMT systems (character alignments) 215
Table 105: HU Test Set 3 results of Urdu to Hindi SMT systems (cluster alignments) 215

Introduction

In general, the term *translation* is understood as the process of understanding the meaning of a text in one language and subsequently producing an equivalent text in another language, conveying the same message. Machine Translation (MT) is a *rêve* of the 1950s [21, 24-26, 80, 125, 171, 172, 178, 179]. Although a large number of milestones have been achieved to enliven the *rêve* of MT [21, 23-26, 28, 32, 41, 80, 87-90, 93, 100-102, 104, 105, 113, 115, 125, 131, 132, 134, 135, 139, 144, 162, 171, 172, 178, 179, 188], it is still a dream in the interdisciplinary research of *computer science*, *artificial intelligence*, *machine learning*, *computational linguistics*, *natural language processing* and *engineering*.

The dream of MT is fuzzy like other dreams. To make it crisp, we have to consider precise translation tasks. General purpose, high quality and fully automatic MT is believed to be impossible [10, 24-26]. But the general MT problem can be reduced to various subproblems obviously less complex and less hard than the general one. We will concentrate on a few of them of particular interest, such as *intralingual* or *interdialectal* translation. That problem reduction can be made on the basis of the domain of application, the sublanguage (a restricted and limited part of a language) considered for translation, the intended users, the language pairs under consideration, *etc.*

These features also help to define the goal and objectives for the subproblems of MT [21, 23-27, 41, 80, 93, 102, 115, 125, 139, 171, 172, 178, 179, 187, 190]. However, these subproblems may still be very hard and complex, although some instances may be quite simple or not so difficult. We will define later more precisely what we mean by complexity and difficulty. In any case, there is a long way ahead to go [25, 26, 139, 187]. One of our goals will be to characterize the complexity and difficulty of solving a certain class of translation problems that we will call "*weak translation problems*".

MT is known for its complex nature and *multivalence* is one of its main reasons. *Multivalence*, the term used by Mel'čuk [125], arises due to the non-determinism (*polysemy* during analysis and *synonymy* in generation, and both in transfer) [6, 13, 21, 23-26, 32, 41, 51, 93, 102, 113, 115, 125, 139, 149, 154, 178, 179, 187]. The number of possible translations of an average source language sentence may go up to thousands or in general increase dramatically with its length [6, 13, 21, 25, 26, 41, 93, 113, 125, 139, 149, 154, 178, 179, 187]. Given a source language SL and a target language TL, a translation unit S in SL of n words may have an exponential number of valid translations $T_1, T_2, \ldots, T_N$ in TL where $N = O(k^n)$ for some $k > 1$ depending on the precise subproblem at hand.

To resolve the problem of *multivalence* for a given subproblem of MT, different filters are applied at various levels during the phases of preprocessing, analysis, synthesis and postprocessing to restrict the cardinality of the possible solution set of the problem to an acceptable and reasonable range of values [6, 13, 21, 23-28, 41, 51, 80, 93, 102, 104, 113, 115, 125, 139, 149, 154, 171, 172, 178, 187].

Transliteration is also a subproblem of MT. It consists in overcoming the scriptural differences among different writing systems used for different languages [1, 3, 4, 9, 15, 16, 47, 50, 57, 59-61, 65, 73, 82-85, 97, 100, 101, 108, 112, 124, 130, 143-146, 150, 153, 165, 168, 174, 181, 189, 191] or even for the same language [118-121, 164].

We are interested in the special class of MT subproblems where N is either very small, say always less than 5, or even almost always equal to 1 because of the proximity of the written forms of SL and TL. For example, this happens in situations (1) when the languages of a translation pair are extremely close to each other, *e.g.* Bengali–Assamese, Hindi–Marathi, Hindi–Urdu,

etc., (2) when translation is performed between two different varieties or dialects of a language, either written in the same writing system (Quebecois–French, Malay–Indonesian) or in multiple unintelligible writing systems (Punjabi, Sindhi, Seraiki) and (3) when the same language is written in different mutually incomprehensible scripts (Kashmiri, Malay, Punjabi, Sindhi, Seraiki).

The domain of our investigation is the class of subproblems π of MT, applied to a pair $\langle(L_i, W_j), (L_k W_l)\rangle$ of combinations of a language and a writing system, such that there exists only one (in most of the cases) or a very small set of valid "translation solutions" to a subproblem π for a given sentence S of L_i written in W_j. A natural assumption is that such problems should be very simple (in terms of complexity of the sufficient computational model) and not very difficult (in terms of the human and computation costs involved in preparing the system to perform translation) than the general translation problems. We will show that the complexity and the difficulty to solve *weak translation problems* can vary considerably.

The complexity and the difficulty of a subproblem π depend on the precise instance of the *weak translation problem*, here denoted by $\pi\ \langle(L_i, W_j), (L_k W_l)\rangle$. We will also use the notation $\pi(SL/SW, TL/TW)$. For example, the complexity and difficulty of interdialectal translation is less for Malay/Latin–Indonesian/Latin than for Hindi/Devanagari-h–Marathi/Devanagari-m[1] translation. We can categorize *weak translation problems* into generic and specific subproblems.

Intralingual localization is a generic problem that can be further refined in the specific problems of *word for word translation* and *intralingual translation* between different varieties of the same language. For example, IBM product documentation in French is translated into French by Bull[2], a French computer company that sells IBM products (e.g. AS4000 under AIX) as OEM. Bull does not use the French versions prepared by IBM because IBM terminology is not identical to Bull terminology[3]. This kind of translation is also mandatory to localize the Quebecois dialect in France, *e.g.* the Quebecois term '*présentement*' must be localized into '*maintenant*' in France and vice versa. Similar problems also exist between English (UK) & English (USA), French of 14th century–standard French, and Malay (Malaysia)–Indonesian (Indonesia).

To solve these problems, a full syntactic analysis is not required, but we have to perform a *word for word translation* for the localization of one variety or dialect of the language into the other and vice versa. Table 1 gives a list of generic and specific subproblems of the general *weak translation problem*, together with some of their instances, in increasing complexity and difficulty order.

Successive generic problems are more complex than the previous ones. For example, the Quebecois–French pair relates to both the first and the third generic problem. In case of intralingual localization, *word for word translation* is sufficient to perform lexical Quebecois–French translation, but we need to do a more complex analysis to perform interdialectal Quebecois–French translation.

[1] The script used for Hindi and Marathi are different variants of the Devanagari script (the original is used for Sanskrit).

[2] We refer here to the documentation of AIX, IBM proprietary version of UNIX http://www.bull.com/index.php

[3] As Hagège said, "languages are the flags of national identity". Here, company terminologies are flags of company identities.

Sr.	Generic Subproblem	Specific Subproblems	Instances	Constraints
1	Language localization or intralingual localization	Word for word translation Intralingual translation	Unix documentation IBM to Bull (French to French)	$SL = TL$ $SW = TW$
			Québécois–French	
			Malay–Indonesian	
2	Scriptural translation	Transliteration Transcription Phonetic transcription	Malay/Latin–Malay/Jawi	$SL = TL$ $SW \neq TW$
			Sindhi/Sindhi[4]–Sindhi/Devanagari	
			Punjabi/Gurmukhi–Punjabi/Shahmukhi	
			French/Roman–French/IPA[5]	
		Transliteration Transcription Phonetic transcription	Hindi–Urdu	$SL \neq TL$ $SW \neq TW$
			Bengali–Assamese	
			Hindi–Marathi	
3	Interdialectal translation	Word for word translation Scriptural translation Intralingual translation	Quebecois–French	$SL = TL$ $SW = TW$
			English (USA)–English (UK)	
			Malay/Latin–Indonesian/Latin	
			Sindhi/Sindhi–Sindhi/Devanagari	$SL = TL$ $SW \neq TW$
			Punjabi/Gurmukhi–Punjabi/Shahmukhi	
			Malay/Jawi–Indonesian/Latin	
4	Bilingual translation	Word for word translation Scriptural translation Bilingual translation between linearly very similar languages	Hindi–Urdu	$SL \neq TL$ $SW \neq TW$
			Bengali–Assamese	
			Hindi–Marathi	

Table 1: Subproblems of the weak translation problem (by order of increasing complexity)

Linguistic Architecture

Following [21, 26, 28, 178, 179, 190], we have adopted and adapted the *framework for syntactic translation*, shown in Figure 1, to solve the *weak translation problems*.

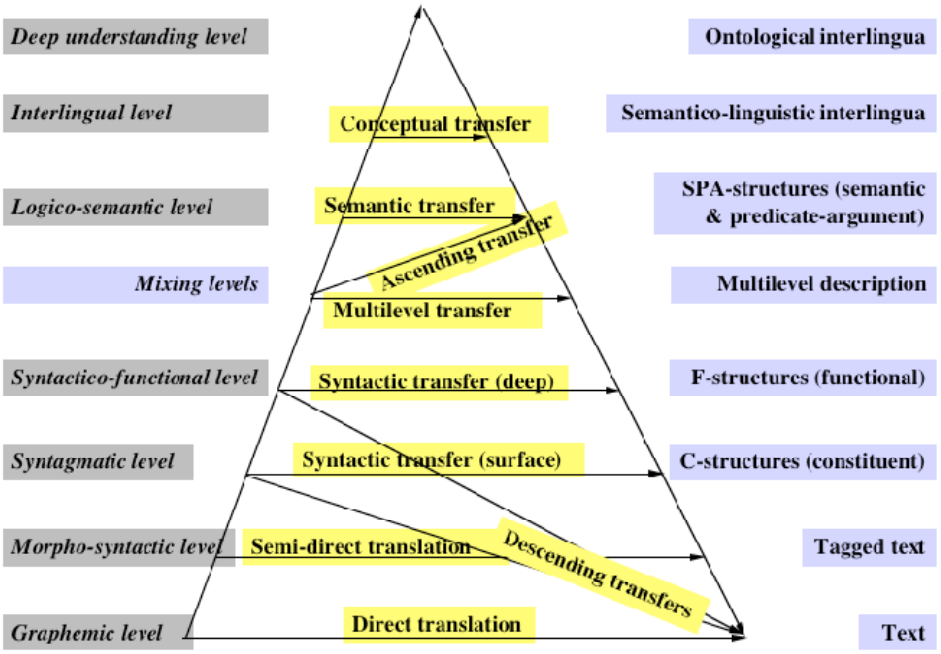


Figure 1: Vauquois' triangle [21, 26, 28, 178, 179]

[4] The derivation of the Perso-Arabic script is known as the Sindhi script.

[5] International Phonetic Alphabet (IPA).

We use *interlingua* and *transfer-based* linguistic architectures, and experiment with expert, empirical and hybrid approaches. We go through various levels of linguistic representation: source (in SW), morphotactic, morphological, morphosyntactic and Universal Intermediate Transcription (UIT). The refinement of Vauquois' triangle to the weak translation problems is shown in Figure 2.

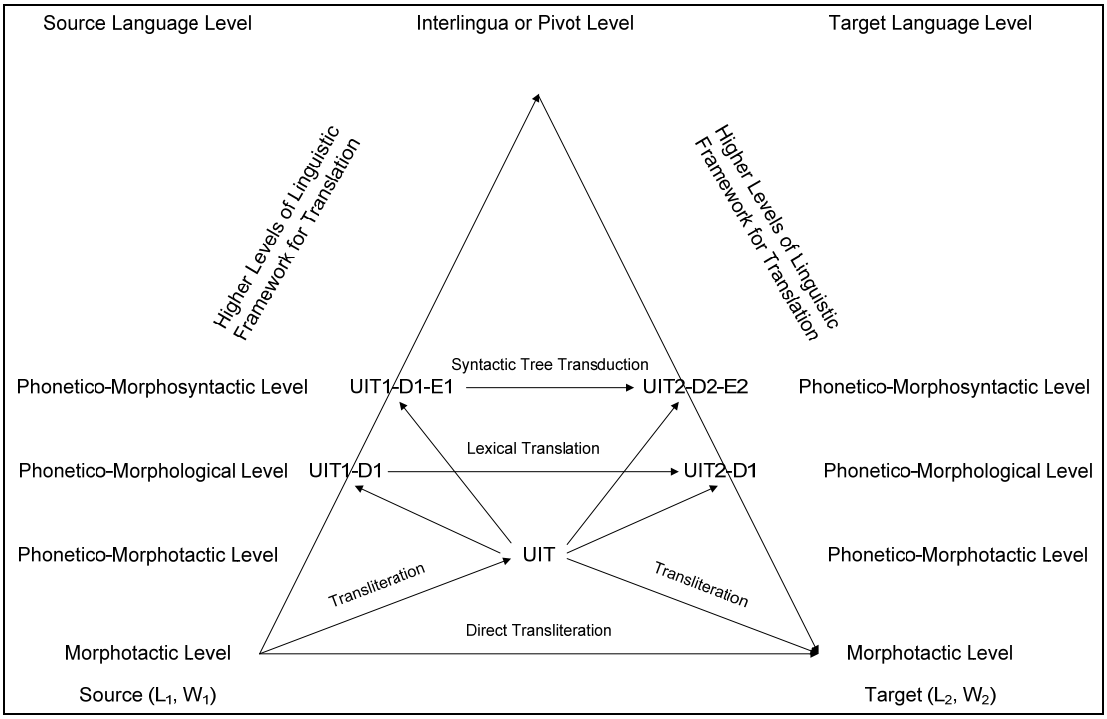


Figure 2: Adopted and adapted Vauquois's triangle for the weak translation problem

UIT is defined for each group of very closely related languages or dialects and serves as a Pivot. More precisely, it is used as a *phonetico-morphotactic* pivot for surface morphotactic translation, as a *phonetico-morphological* pivot for word for word translation, and as a *phonetico-morphosyntactic* lexical pivot for syntax-based translation (in conjunction with syntactic transfer).

Computational Model

Researchers have employed various computational models ranging from *finite-state technology* [131-135, 161, 162, 166, 167, 169] to *machine learning*, and *empirical methods* [1, 32, 33, 41, 102, 104, 115] for solving different subproblems of MT.

Following [3, 4, 100, 101, 134, 135, 162, 174], we employ *finite-state technology* for solving different subproblems of the weak translation. For example, we use non-probabilistic finite-state transducers [121] to solve the problem of *scriptural translation* (we will define that term precisely later). We also use *Context-Free Grammar* (CFG) for developing "phrase structure grammars".

Finite-state methods give a 16.1% *word error rate* for Urdu to Hindi *scriptural translation* when all necessary information is present in the input text (we will explain later what we mean by necessary information). They give a 47% *word error rate* when all necessary information is not present in the input text (the usual and normal case especially for Urdu to Hindi, Punjabi/Shahmukhi to Punjabi/Gurmukhi, *etc*.). At sentence level, the finite-state methods give a 90% *sentence error rate* for Urdu to Hindi scriptural translation when the input Urdu text contains the required information. Without the required information in the input Urdu text, they give a 99% sentence error rate.

Due to the successful use of SMT models in MT, we conjectured that SMT could give us better results than our finite-state model. Indeed, SMT decreases the *word error rate* from 47% to 22.1% for Urdu to Hindi transliteration when the input Urdu text does not contain the diacritical

marks (mandatory for performing Urdu to Hindi *scriptural translation*). At sentence level, it decreases the error rate from 99% to 95%. In contrast, our finite-state model gives better results (16.1% word error rate) than our best SMT model (27.8% word error rate) when the Urdu text does contain all necessary diacritical marks.

The absence of information in the source side, which is the usual case for Urdu to Hindi scriptural translation, is a big challenge and cannot be handled well within the framework of *non-probabilistic finite-state transducers*. Although SMT increases the word accuracy in such cases, the results are still not satisfactory as far as usability in real context is considered.

To increase the accuracy, we have proposed a *hybrid model* [120] for *scriptural translation* and gained an overall accuracy of 79.1% (word-level) when the input Urdu text does not contain the diacritical marks. A hybrid model, a combination of finite-state and statistical models, gives better results than the previous two models. In short, we have employed finite-state, empirical and statistical, context-fee grammars, tree transduction and syntax-based translation (in conjunction with syntactic transfer) to solve different generic and specific subproblems of *weak translation problems*. Table 2 shows results of Urdu to Hindi scriptural translation of different approaches used to solve the problem of Urdu to Hindi scriptural translation on the same test set.

Approach	**Word Error Rate**		**Sentence Error Rate**	
	With information	*Without information*	*With information*	*Without information*
FST	16.1%	47%	90%	99%
SMT	27.8%	23%	94.5%	95%
Hybrid	14.2%	20.9%	86%	93%

Table 2:Results of Urdu toHindi scriptural translation

Evaluation Methods

One of the most difficult things in MT is the evaluation of a proposed system/algorithm. A *natural language* is not an object of exact science like Mathematics or Physics. Therefore, the understanding of a natural language is a subjective problem that depends on multiple factors. For example, *multivalence* makes it hard to associate a real objective number to an MT evaluation.

Recent MT evaluation campaigns have been criticized because only tables of figures (such as BLEU, NIST, ORANGE, METEOR…) are shown as results, while these n-gram based measures have been shown not to correlate very well with human judgments [40]. Commercial MT systems have been consistently ranked low by these measures, while human judges ranked them quite high [81]. We also have achieved an average 80% word accuracy for Hindi-Urdu scriptural translation with our finite-state methods that seem to be a good measure. But if we measure the accuracy at the sentence level, then we have an accuracy of 1% to 14%. Thus, it is important to do subjective evaluations in addition to the objective evaluations.

In general, *human* and *machine* (automatic) evaluation methods are used to evaluate an MT system. Human evaluations of MT judge various aspects of translations, including adequacy, fidelity and fluency [72, 185]. They are relevant, but costly in terms of time and money [72, 147]. For automatic evaluations, there exist different evaluation metrics like BLEU [147], NIST [49], F-measure [177], METEOR [111], *Word Error Rate* (WER) [138], and MaxSim [43]. Finally, there are objective task-based metrics measuring human performance like post-editing time and time to goal (e.g. time to perform a booking, a spoken dialogue translation system). Mostly, we have used n-gram based *automatic* evaluation methods, but also have used *subjective human* evaluation while the post editing for achieving a perfect result.

We have evaluated the quality of our results on different aspects like *Sentence Accuracy* (SA), *Word Accuracy* (WA), *post-editing time*, *confidence level of human evaluator*, *fluency*, *adequacy* and *usability*. We have devised scales for each aspect used for categorizing different systems and measuring their translation quality. For example, the scale devised for SAR is formulated in Table 3. We will discuss and formulate all these scales later in more details in the sequel.

Scale Point	Relation with SAR	Description
0	SAR < 5%	NULL
1	5% ≤ SAR < 10%	OK
2	10% ≤ SAR < 15%	AVERAGE
3	15% ≤ SAR < 25%	GOOD ENOUGH
4	25% ≤ SAR < 50%	GOOD
5	50% ≤ SAR ≥ 70%	VERY GOOD
6	SAR > 70%	EXCELENT

Table 3: Scale based on sentence accuracy for scriptural translation quality

N-gram co-occurrence automatic scoring metrics like BLEU and NIST are widely used as benchmark for MT system evaluation especially BLEU, even with its known shortcomings for evaluation of general MT systems [40]. We will show that these deficiencies are not that significant in the case of the evaluation of *weak translation problems*, because we have a unique or very small number of references, say 2 to 4. Thus these metrics are good measures for the translation quality of *weak translation problems*. We have used BLEU and NIST to evaluate our translation systems.

Thesis Plan

This report is mainly divided into three parts. The first part introduces the weak translation problems. The first chapter introduces and gives an analysis of scriptural translation problem. In the second chapter, we describe the finite-state approach for solving scriptural translation problems. Finally, we report the results of our finite-state approach on the Indo-Pak languages (the term is explained later).

The second part also consists of two chapters. In the third chapter, we describe our experiments for solving Hindi–Urdu scriptural translation problem using Statistical Machine Translation (SMT) and report our results. The last part of this chapter, we illustrate our hybrid approach (a novel combination of finite-state and statistical approaches) for solving the scriptural translation problems for the Indo-Pak languages. In the fourth chapter, we describe our interactive scriptural translation model. We also describe our evaluation methodologies for the interactive scriptural translation systems.

The third and final part consists of an analysis of *interdialectal machine translation*, a higher-level weak translation problem that requires more complex analysis and computation approaches than the scriptural translation problems. In this section, we analyze different computation approaches and required resources for solving the interdialectal translation problem for the Indo-Pak languages. Finally, we conclude our work and give future perspectives of our study.

Part I
Scriptural Translation and other Weak Translation Problems

Chapter 1. Scriptural Translation

Transliteration or transcription, a subproblem of general translation, is vital for Natural Language Processing (NLP), especially in the domains of Machine Translation (MT), Cross-Lingual Information Retrieval (CLIR), Named Entity Recognition (NER), multilingual text and speech processing. It is also known as *translation-by-sound*. In this chapter, we throw light on the importance of transliteration and/or transcription in NLP and its related fields and briefly explain various transliteration approaches. We introduce, define and explain the term *scriptural translation*. Finally, we give a brief account of different techniques for implementing scriptural translation.

1.1. Scriptural Translation, Transliteration and Transcription

The terms transliteration and transcription are often used as generic terms for various processes like transliteration, transcription, romanization, transcribing and technography [66]. Transliteration is defined as "to write a word or letter in a different alphabet"[6]. It denotes a process that maps one writing system into the other, ideally letter by letter. It attempts to use a one-to-one grapheme correspondence (orthographic conversion). A good transliteration is a reversible process to ensure that the source word can be regenerated from the target transliterated word [66]. On the other hand, transcription is defined as "a written representation of words or music". In the words of [66], "Transcription is the representation of the source script of a language in the target script in a manner that reflects the pronunciation of the original, often ignoring graphemic (character-to-character) correspondences."

In general, the speech processing community uses the term transcription to denote the process of conversion from the script or writing system to the sound (phonetic representation). For example, the transcription of the word "love" in the International Phonetic Alphabet (IPA) is [ləv]. On the other hand, the text processing community uses the term transliteration and defines it as the process of converting a word written in one writing system into another writing system while preserving the sound of the original word [1, 3, 4, 9, 101, 145, 146, 174]. More precisely, the text processing community defines the term transliteration as two transcription processes "source script to sound transcription" and "sound to target script transcription" and sometimes as one transcription process "source script to target script transcription". We propose the new term *scriptural translation* for this combined process. Scriptural translation is the process of transcribing a word written in the source language script into the target language script by preserving its articulation in the original language in such a way that the native speaker of the target language can produce the original pronunciation.

In the remaining sections, we will use the term transliteration as it is used in the literature.

1.2. Machine Transliteration in Natural Language Processing (NLP)

Transliteration is crucial for handling Out-Of-Vocabulary (OOV) words, proper nouns and Named Entities (NE) in various fields of NLP, especially in MT, CLIR, multilingual text and speech processing and development of multilingual resources and applications. Back–transliteration refers to the process of retrieving the original word in the source language from the transliterated word in the target language. Generally, transliteration compromises on the

[6] Word definitions are taken from Cambridge Advanced Learner's Dictionary.

source language information while approximating the pronunciation of the original source word into the target language. This makes back–transliteration a more challenging and complex problem.

Transliteration refers to phonetic translation across two languages with different writing systems [100, 101], such as Arabic to English [1, 3, 4, 9, 168, 174]. Most prior work on transliteration has been done for MT of English, Arabic, Japanese, Chinese, Korean, etc., [1, 3, 4, 9, 15, 16, 50, 65, 83, 85, 97, 100, 101, 130, 174, 191] for CLIR [57, 82, 112, 150, 165, 180, 181, 189], and for the development of multilingual resources [84, 189].

Various transliteration models have been proposed by several researchers: *grapheme-based* models [1, 50, 60, 61, 65, 73, 82, 84, 85, 97, 112, 130], *phoneme-based* models [59, 83, 100, 101, 129, 174, 180, 181], *hybrid* models [3, 4, 9, 15, 16] and *correspondence-based* models [143, 144, 146]. *Grapheme-based* models establish direct correspondences between graphemes of the source language and the target language for the purpose of transliteration. On the other hand, *phoneme-based* models use the source language phonetic knowledge and its phonemes as the pivot to perform transliteration. *Hybrid* and *correspondence-based* models use both graphemes and phonemes to transliterate the source text into the target text.

In general, all transliteration models exploit different methods like Weighted Finite-State Transducers (WFST), Machine Learning algorithms (Maximum Entropy, Decision Trees, Memory-based learning), statistical methods and Statistical Machine Translation (SMT), etc. for transliterating the source into the target to handle proper names, OOV words, technical terms, Named Entities (NE), etc. in various domains of NLP [145].

1.2.1. Grapheme-based models

Grapheme-based models directly convert the source language graphemes into the target language graphemes without relying on the knowledge of source language phonetics [65, 130, 146]. They develop direct mappings between graphemes of the source and the target languages. Various transliteration approaches have been proposed based on the *source channel* model [82, 112], the *joint source channel* model [50, 65, 130], the *SMT* model [1, 73, 97, 124], the *decision tree* model [84] and the *transliteration network* model [61, 85, 145].

Direct orthographic mapping from English to Chinese is a very difficult task due to the large number of characters and multiple orthographic variants of Chinese [66, 67]. For example, the English name ‘Smith’ can be segmented into /s-mi-th/ but there exist three Chinese characters for each of /s-/, /-mi-/ and /-th/, given in Table 4 [65]. The best transliteration is shown in bold.

English	/s-	-mi-	-th/
Chinese 1	**史**	**米**	**斯**
Chinese 2	斯	密	史
Chinese 3	思	麦	瑟

Table 4: English-Chinese direct orthographic mapping example [65]

Both [65] and [130] have reported 45.6% and 47% word error rates for English to Chinese transliteration using the *n-gram transliteration* model and the *n-gram noisy channel* model respectively. [50] have reported a *word agreement ratio* of 69.3% and 67.9% for transliteration from Bengali to English and vice versa respectively, using their *modified joint source channel* model. [97] have reported that the correct transliteration of a word is generated in more than 76% of the cases. [73] has reported 51.08% and 56% of accuracy for transliteration from Chinese to English using a *character-based cluster-specific* model and a *phrase-based cluster-specific* model respectively.

All these reported results are at word or phrase level because they are treating certain types of words like proper nouns, OOV words, technical terms and Named entities. They have not given any evaluation at sentence level or any subjective evaluation.

1.2.2. Phoneme-based models

Phonetic knowledge of the source language is the basis of the *phoneme-based* model. In this model, a source language grapheme is first converted into a source language phoneme and then the source language phoneme is converted into the target language grapheme. [101] proposed a *phoneme-based* model for Japanese to English transliteration using WFSTs. They divided the problem into a number of subproblems, *e.g.* English word probabilities, English word to pronunciation, English pronunciation to Japanese pronunciation, *etc*. They developed a WFST for each subproblem, and finally composed these WFSTs to perform the complete task of back–transliteration from Japanese to English. They reported a 64% word accuracy. [174] developed a similar model for Arabic to English transliteration and reported approximately 33% success. A similar Chinese to English transliteration model [129] was proposed using different parameters for different steps for performing *spoken document retrieval*. An extended Markov window method was proposed for English to Korean transliteration and a 54.9% word accuracy was reported [83].

[180] developed a *phoneme-based* transliteration model for English to Mandarin Chinese using WFSTs to perform IR. They first generated the English pronunciation using the Festival[7] speech synthesis system. Then, the English pronunciation was converted into a sequence of *generalized initials and finals* (GIFs) and then this GIFs sequence was transformed into the *pin-yin* (Latin-based) transcription, but without tone. Finally, they performed the translation of *pin-yin* sequence to Chinese character sequence. They developed deterministic transformations and statistical transformations. The whole process is shown in Figure 3.

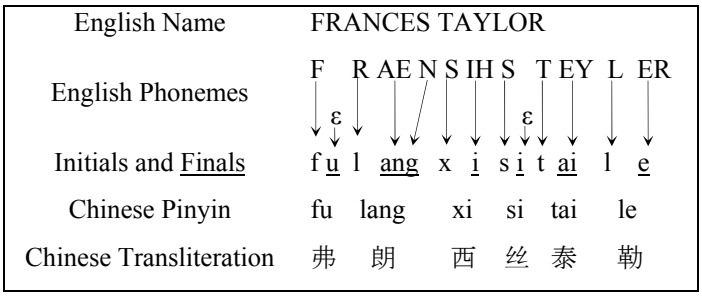


Figure 3: Four steps in English-to-Chinese transliteration of names [180]

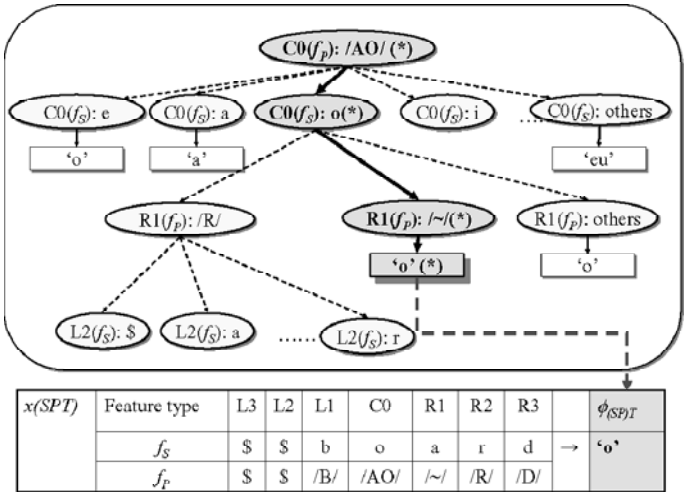
1.2.3. Hybrid and correspondence-based models

Hybrid models try to use both graphemes and phonemes mappings for transliterating between the source and the target languages. [3] linearly combined a *phoneme-based* model and a *spelling-based* model to perform Arabic to English transliteration. Transliteration results were improved by spelling correction on the source side and by applying web-based filtering on the target side. They achieved a word accuracy of approximately 50%. [4] exploited comparable corpora and described a two-step method for Named Entities (NE) transliteration with a top-1 word accuracy of 72.57%. First, they generated a ranked list of transliteration candidates using bilingual and monolingual resources and then they rescored the list of candidates using different monolingual cues. [16] and [15] linearly combined the *grapheme-based* and *phoneme-based* models to perform transliteration between English and Japanese and reported approximately 60% word accuracy.

[143] considered both the context of a source character and its corresponding pronunciation for English to Korean transliteration. [146] described a *correspondence-based* model using different machine learning algorithms (a) *maximum entropy* method, (b) *decision tree* and (c) *memory-based learning*. Their method used two component functions Φ_{SP} and $\Phi_{(SP)T}$. The Φ_{SP} function produced correspondences between the source grapheme and source phoneme and the

[7] http://www.speech.cs.cmu.edu/festival

$\Phi_{(SP)T}$ function (S×P→T) produced target graphemes corresponding to both the source grapheme and the source phoneme. Figure 4 shows part of a decision tree constructed for $\Phi_{(SP)T}$ in English to Korean transliteration. They reported 62%, 63.3% and 66.9% success for English to Korean transliteration and 66.8%, 76% and 72.2% success for English to Japanese using the *decision tree*, *maximum entropy* and *memory-based learning* methods, respectively.



x(SPT)	Feature type	L3	L2	L1	C0	R1	R2	R3		$\phi_{(SP)T}$
	f_S	$	$	b	o	a	r	d	→	'o'
	f_P	$	$	/B/	/AO/	/~/	/R/	/D/		

Figure 4: Part of the decision tree for $\Phi_{(SP)T}$ [146]

None of the above described works have done any subjective evaluation like the usability and effectiveness of their systems in the real scenarios. They have reported good percentages for word and phrase accuracies, but we cannot easily correlate these percentages with subjective measures like usability and effectiveness of results and user satisfaction. For example, we have achieved 83.9% word accuracy with our finite-state methods for scriptural translation, but at sentence level we have only 10% accuracy. According to our scale, we can classify our system in the OK class. Although, we have a very high word level percentage, our system is only classified as OK. Thus, in addition to objective evaluation, we have also considered the subjective evaluation because we want to develop online services that people can use in their daily life.

1.3. Scriptural Translation

Scriptural Translation is a subproblem of general translation and almost always a *weak translation problem*. It is a special kind of process of transcribing a word written in the source language script into the target language script by preserving its articulation in the original language irrespective of the word type in such a way that the native speaker of the target language can produce the original pronunciation. Solving the scriptural translation problem is vital to bridge the scriptural divide between speakers of the same language.

There are many languages that are written in two or more mutually incomprehensible scripts. For example, Punjabi is written in three different scripts: Shahmukhi (a derivation of the Persio-Arabic script), Gurmukhi and Devanagari. Kazakh and Kurdish are also written in three different scripts, Arabic, Latin and Cyrillic. Malay has two writing systems, Latin and Jawi (a derivation of the Arabic script), *etc.* In Europe, Serbo–Croatian is written in the Latin script chiefly in Croatia, in the Arabic alphabet mostly in Bosnia and in the Cyrillic and Latin scripts in Serbia [114].

Figure 5 shows some languages that are written in two or more mutually incomprehensible scripts, with the number of their native speakers in millions. In Figure 5, the example phrase 'All people' is written in various scripts for each language and its pronunciation in IPA is also given between brackets at the end. Arrow signs show the direction of the writing systems (left-to-right or right-to-left).

Kashmiri (5.6 M)	Kazakh (8.3M)	Kurdish (16 M)	Sindhi (24.2M)
یوزم سونٛتک बऊज़ुम सौंतुक [buʤum soɲʈuk]	Barlyq adamdar بارلیق ادامدار Барлық адамдар [barljq əd̯amd̯ar]	Hemû mirov هیموو مروو Cyrillic to add [hemvu mɪrov]	هر اِنسان हर इंसान [hər ɪnsan]
Punjabi (106.5M)	**Malay (39.1M)**	**Serbo–Croatian (18.5M)**	**Saraiki (14M)**
سارے اِنسان ਸਾਰੇ ਇਨਸਾ सारे इंसान [sare ɪnsan]	Semua manusia سموا مأنسي [səmuə maɲsɪə]	Sva ljudska Свag људска سوا لودسکا [svə ludskə]	सारे इंसान سارے اِنسان ਸਾਰੇ ਇੰਸਾਨ [sare ɪnsan]

Figure 5: Languages with different writing systems[8]

There are dialects of the same language that are written in mutually unintelligible scripts, like Punjabi, Malay and Sindhi. Some examples are shown in Figure 5. There also exist pairs of closely related languages, written in mutually incomprehensible writing systems. For example, Hindi is written in Devanagari and Urdu is written in a derivation of the Persio-Arabic script. Other examples are Bengali–Assamese and Hindi–Marathi. Figure 6 shows an example sentence of Hindi–Urdu.

دنیا کو امن کی ضرورت ہے۔

दुनिया को अमन की ज़रूरत है।

[d̪ʊnɪjə ko əmən ki zərurət̪ hæ.]

The world needs peace.

Figure 6: Hindi - Urdu example sentence

Scriptural translation is an endeavor to bridge the scriptural, ethnical, cultural and geographical divisions among various communities around the world. It is a prerequisite for weak translation problems like *interdialectal translation*, *translation between closely related languages*, *etc.* without lexical transformation, when the source and the target language dialects or language pairs are written in mutually unintelligible writing systems. Examples are shown in Figure 5 and 6.

It differs from general transliteration in various aspects. Generally, transliteration handles only OOV words, proper nouns, NEs, *etc*. On the other hand, *scriptural translation* must handle all kinds of words irrespective of their type. It provides basis for *Cross-Scriptural Machine Translation* (CSMT), *Cross-Scriptural Information Retrieval* (CSIR), *Cross-Scriptural Application Development* (CSAD), *Inter-Dialectal Machine Translation* (IDMT), *Cross-Dialectal Information Retrieval* (CDIR) and for solving the *weak translation problems*.

It is a generic subproblem of the general translation problem and of the weak translation problem. Table 5 gives a list of specific subproblems and instances of scriptural translation within the broader categories of general translation problems and weak translation problems. In each broader category, the problems and instances are listed in an increasing order of complexity and difficulty.

[8] Some sample phrases are taken from http://www.omniglot.com/

Sr.	General	Generic	Specific	Instances	Constraints
1	General translation subproblems	Scriptural translation	Transliteration Transcription Phonetic transcription	English–French French–Spanish French–Italian	$SL \neq TL$ $SW = TW$
			Transliteration Transcription Phonetic transcription	French–Arabic English–Arabic English–Japanese English–Korean English–Chinese *etc*.	$SL \neq TL$ $SW \neq TW$
2	Weak translation subproblems	Scriptural translation	Transliteration Transcription Phonetic transcription	Malay/Latin–Malay/Jawi	$SL = TL$ $SW \neq TW$
				Sindhi/Sindhi–Sindhi/Devanagari	
				Punjabi/Gurmukhi–Punjabi/Shahmukhi–Punjabi/Devanagari	
				French/Roman–French/IPA	
				Seraiki/Shahmukhi–Seraiki/Devanagari–Seraiki/Gurmuhi	
				Kashmiri/Urdu[9]–Kashmiri/Devangari	
			Transliteration Transcription Phonetic transcription	Hindi–Urdu	$SL \neq TL$ $SW \neq TW$
				Bengali–Assamese	
				Hindi–Marathi	

Table 5: Subproblems of scriptural translation for general translation and weak translation

Different challenges and barriers for the scriptural translation are sorted into different categories:

- Challenges and barriers within the same language (Malay/Latin–Malay/Jawi, Kashmiri/Urdu–Kashmiri/Devanagari).
- Challenges and barriers between dialects of the same language (Punjabi/Shahmukhi–Punjabi/Gurmukhi, Sindhi/Sindhi–Sindhi/Devanagari).
- Challenges and barriers between closely related languages (Hindi–Urdu, Hindi–Marathi).

Some challenges are common to all these categories. Now we will discuss each category separately.

1.3.1. Challenges and barriers within the same language

In this section, we describe the main challenges and barriers for the scriptural translation problems of the same language, written in two or more different scripts.

1.3.1.1. Scriptural divide

There exists a written communication gap between people who can understand each other verbally but cannot read each other. They are virtually divided and become *scriptural aliens*. Examples are the Hindi & Urdu communities, the Punjabi/Shahmukhi & Punjabi/Gurmukhi communities, *etc.* Examples, showing the scriptural divide, are shown in Figure 5 and Figure 6. Such a gap also appears when people want to read some foreign language or access a bilingual dictionary and are not familiar with the writing system. For example, Japanese–French or French–Urdu dictionaries are useless for French learners because of the scriptural divide. The

[9] Kashmiri is written in Urdu script, a derivation of the Perso-Arabic script, with few additions

author has faced the same problem when he started to learn French in Pakistan in 2005[10]. Table 6 gives some figures on how this scriptural divide affects a large population of the world.

	Native Speakers	2nd Language Speakers	Total
Hindi	366,000,000	487,000,000	853,000,000
Urdu	60,290,000	104,000,000	164,290,000
Total	426,290,000	591,000,000	**1,017,290,000**
Source: (Grimes, 2000) all numbers are in millions			

Table 6: Population of Hindi and Urdu [156]

Punjabi is also the native language of more than 110 million people of Pakistan and India [119]. Hindi, Urdu and Punjabi represent a total population of 1,127 million, more than the total population of Europe (831 million in 2009)[11].

1.3.1.2. Under-resourced and under-written languages

Under-resourced and under-written features of the source or target language are the second big challenge for *scriptural translation*. The lack of standard writing practices or even the absence of a standard code page for a language makes transliteration or transcription very hard. The existence of various writing styles and systems for a language leads towards a large number of variants and it becomes difficult and complex to handle them.

For example, Lingala, a *Bantu* language group of the *Niger-Congo* family, is more a spoken language than a written language, and has several different writing systems with 10 million native speakers, mainly in the two Congo states. Due to the low literacy of Lingala speakers in Lingala (the literacy rate in Lingala as a first language is between 10% to 30%), its popular orthography is not standardized and varies from one region of Congo to the other[12].

Similarly, Punjabi is the largest language of Pakistan (more than 70 million) and is also more a spoken language than a written one. There existed only two magazines (one weekly and one monthly) in 1992 [155]. In the words of [156], "… *there is little development in Punjabi, Pashto, Balochi and other languages*…". [117] reports the first effort towards establishing a standard code page for Punjabi-Shahmukhi and till date, a standard code page for Shahmukhi does not exist. Similar problems also exist for the Kashmiri and Seraiki languages.

1.3.1.3. Absence of necessary information

There are cases where the necessary and indispensable information for scriptural translation are missing in the source text. For example, the first word دنيا [ɖʊnɪjɑ] (world) of the example sentence of Figure 6 misses crucial diacritical information, mandatory to perform Urdu to Hindi scriptural translation. Like in Arabic, diacritical marks are part of the Urdu writing system but are sparingly used in writings [120, 121, 194].

Figure 7(a) shows the example word without diacritical marks and its wrong Hindi conversion according to conversion rules (explained later). The Urdu community can understand the word in its context or without the context because people are tuned to understand the Urdu text or word without diacritical marks, but the Hindi conversion of Figure 7(a) is not at all acceptable or readable in the Hindi community. On the other hand, Figure 7(b) shows the example word with diacritical marks and its correct Hindi conversion, readable by the Hindi community, according to conversion rules. Similar problems arise for Punjabi/Shahmukhi–Punjabi/Gurmukhi, Sindhi/Sindhi–Sindhi/Devanagari, Seraiki/Shahmukhi–Seraiki/Devanagari, *etc.* Missing infor-

[10] In some learning books, French was transcribed in the Urdu script instead of Roman to ease the process of learning for Urdu speaking community, but I could not use the Urdu – French dictionaries directly due to the non-familiarity with the French writing system.

[11] http://en.wikipedia.org/wiki/Demographics_of_Europe

[12] http://en.wikipedia.org/wiki/Lingala

mation in the source text makes the scriptural translation problem computationally complex and difficult.

دنیا = د [ḍ] ن [n] ی [j] ا [ɑ] दनया = द [ḍ] न [n] य [j] ा [ɑ]	دُ نِیا = د [ḍ] ُ [ʊ] ن [n] ِ [ɪ] ی [j] ا [ɑ] दुनिया = द [ḍ] ु [ʊ] न [n] ि [ɪ] य [j] ा [ɑ]
(a) without necessary information	(b) with necessary information

Figure 7: Example of missing information from the source side

1.3.1.4. Different spelling conventions

Different spelling conventions exist across different scripts used for the same language or for different languages because users of a script are tuned to write certain words in a traditional way. For example, the words یہ [je] (this) = ی [j] + ہ [h] and وہ [vo] (that) = و [v] + ہ [h] are used in Urdu and Punjabi/Shahmukhi. The character ہ [h] produces the vowel sounds [e] and [o] in the example words respectively. On the other hand, a word is exactly written as it is pronounced in Devanagari and Gurmukhi. The example words are written as ये [je] & वो [vo] and ਯੇ [je] & ਵੋ [vo] respectively in Devanagari and Gurmukhi. There exist a large number of such conventions between Punjabi/Shahmukhi–Punjabi Gurmukhi, Hindi–Urdu, *etc.* More details are given in annexes 1 to 4.

Different spelling conventions are also driven by different religious influences on different communities. In the Indian sub-continent, Hindi is a part of the Hindu identity, while Urdu is a part of the Muslim identity[13] [155, 157]. Similarly, Punjabi/Shahmukhi and Punjabi/Gurmukhi are parts of the Muslim and the Sikh identities. Hindi or literary Hindi tries to derive its vocabulary from Sanskrit, while Urdu borrows its literary and scientific vocabulary from Persian and Arabic. Hindi and Urdu not only borrow from Sanskrit and Persian/Arabic, but also adopt the original spellings of the borrowed word due the sacredness of the original language. These differences make scriptural translation across scripts, dialects or languages more challenging and complex.

For example, short vowels ि [ɪ] and ु [ʊ] come at the end of some Hindi and Punjabi/Gurmukhi words of Sanskrit origin, but these short vowels ِ [ɪ] and ُ [ʊ] can never occur at the end of a word in Urdu and Punjabi/Shahmukhi [120, 121]. The first short vowel [ɪ] is also used to construct a compound word in Urdu and Punjabi/Shahmukhi, a compounding device borrowed from Persian. Figure 8 shows some Hindi words of Sanskrit origin with short vowels at the end and their Urdu transcriptions.

ि [ɪ]	व्यक्ति–ویکتی (person) [vjəkti], संस्कृति–سنسکرتی (culture) [səɲskrəṯi]
ु [ʊ]	किन्तु–کنٹو (but) [kɪnṯu], धातु–دھاٹو (metal) [ḍaṯu]

Figure 8: Hindi examples with short vowels at the end

While doing scriptural translation from Hindi to Urdu, short vowels ि [ɪ] and ु [ʊ] at the end of a word are converted into long vowels ی [i] and و ُ [u] respectively. But in back transliteration, this approximation increases the complexity of scriptural translation. More details on different spelling conventions are given in annexes in which we discuss different scripts in detail.

[13] The Hindi movement of the late 19th century played a central role in the ideologization of Hindi. The movement started in reaction to the British Act 29 of 1837 by which Persian was replaced by Hindustani/Urdu, written in Persian script, as the official vernacular of the courts of law in North India. It is the moment in history, when Hindi and Urdu started to emerge as Hindu and Muslim identities.

1.3.1.5. Transliterational or transcriptional ambiguities

Character level scriptural translation across different scripts is ambiguous. For example, the Sindhi word اِنسان [ɪnsan] (human being) can be converted into Devanagari either as इंसान [ɪnsan] or इसान* [ɪsan] (* means wrong spellings). The transliteration process of the example word from Sindhi to Devanagari is shown in Figure 9(a). The transliteration of the third character from the left, Noon (ن) [n], is ambiguous because in the middle of a word, Noon may represent a consonant [n] or the nasalization [ɲ] of a vowel.

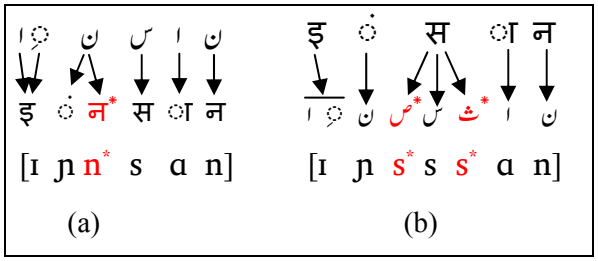


Figure 9: Sindhi Transliteration Example

In the reverse direction, the Sindhi Devanagari word इंसान [ɪnsan] can be converted into a set of possible transliterations [اِنسان, اِنصان*, اِثان*]. All these possible transliterations have the same pronunciation [ɪnsan] but have different spellings in the Persio-Arabic script, as shown in Figure 9(b). Similar kinds of ambiguities also arise for other pairs of scripts, dialects or languages. Thus these ambiguities increase the complexity and difficulty of *scriptural translation*.

1.3.2. Challenges and barriers within dialects

Scriptural translation between different writing systems used for two dialects of a language is a more difficult problem. It possesses all complexity of the previous category:

- Script divide
- Under-resourced and under-written languages
- Absence of necessary information
- Different spelling conventions
- Transliterational or transcriptional ambiguities

Additionally, it possesses other complexities, described below.

1.3.2.1. Distinctive sound inventories

Sound inventories across dialects or languages can be different. Consider the English–Japanese pair. Japanese make no distinction between the 'L' [l] and 'R' [r] sounds so that these two English sounds collapse onto the same Japanese sound [101]. A similar compromise must be done for English 'H' [h] and 'F' [f]. Consequently, the English words *bass*, *bath* and *bus* are transliterated into バス "basu" [b(ə/ɑ)su] [15]. In case of the English Arabic pair, the English 'P' [p] and 'B' [b] are both converted into Arabic 'ب' [b] and in the reverse direction, the Arabic 'ح' [ḥ] and 'ه' [h] are collapsed onto the English 'H' [h] [3]. Other similar cases are shown in Table 7.

English Sounds	Arabic Sounds
D [ɖ]	Deleted [174]
K [k]	ك [k], ق [q] [174]
S [s]	ث [θ], س [s], ص [sˤ] (AbdulJaleel and Larkey, 2003)

Table 7: Examples from English and Arabic distinctive sound inventories

A similar kind of comparison of Japanese and Korean vowels is shown in Table 8. The difference in the vowel systems of Japanese and Korean makes their transliteration a tricky and computationally hard task.

Japanese	Korean	
a as in f*a*ther e as 'ay' in s*ay* i as in mach*i*ne o as in s*o* u as 'oo' in b*oo*k	a [ɑ] e [e] i [i] o [o] u [u]	As in Japanese
	è eu	As in French
	ŭ as 'u' in b*u*t ă as 'a' in Thom*a*s	

Table 8: Japanese Korean vowel comparison

For *Indo-Pak* languages[14], Punjabi/Gurmukhi (a dialect of Punjabi spoken in India) possesses two additional sounds/characters than Punjabi/Shahmukhi (a dialect of Punjabi spoken in Pakistan). Similarly, Hindi, Punjabi, Sindhi and Seraiki have the retroflex form [ɳ], but Urdu and Kashmiri do not. Marathi has 14 vowels in contrast to Hindi's 11 vowels, shown in Table 9.

Hindi Vowels	अ [ə] आ [ɑ] इ [ɪ] ई [i] उ [ʊ] ऊ [u] ऋ [r̥] ए [e] ऐ [æ] ओ [o] औ [ɔ]
Marathi Vowels	अ [ə] आ [ɑ] इ [ɪ] ई [i] उ [ʊ] ऊ [u] ऋ [r̥] ए [e] ऐ [æ] ओ [o] औ [ɔ] अं [əŋ] अः [əh] ऌ [l̥]

Table 9: Hindi and Marathi vowel comparison

Scriptural translation approximates the pronunciation of the source language or dialect in the target language or dialect due to differences between sound inventories across languages or dialects. It is not possible to preserve all source sound distinctions while doing such approximation. Thus a distinctive sound inventory across scripts, dialects or languages increases ambiguities and adds to the complexity of the *scriptural translation* problem. Note that it remains a weak problem because, at sentence-level, there is almost always only one possible solution in the context.

1.3.2.2. Lexical divergence and translational ambiguities

For scriptural translation, one needs to decide that a word will be transliterated or translated. In the case of Bengali–English transliteration, the words জনতা দল [ʤənəţɑ dəl] (a proper name) is transliterated into 'JANATA DAL', although জনতা [ʤənəţɑ] (people) and দল [dəl] (group) are common nouns. On the other hand, যাদবপুর বিশ্ববিদ্যালয় [jɑḍəvpur vɪʃvvɪḍjɑləjɔ], the first word (name of a city) is transliterated into 'JADAVPUR' and the second word is translated to 'UNIVERSITY' [50]. Similarly, in the case of Hebrew to English translation, person names are always transliterated, although many of them have homographs that can be translated. On the other hand, names of countries may be subject to translation or transliteration, *e.g.* [15]צרפת [ərfət] is translated to 'France', while קונגו [kongo] is transliterated into 'Congo' [97].

In case of translations of dialects of the same language or closely related languages, we have to decide that a word will be translated or transliterated. As mentioned above, Hindi derives its vocabulary from Sanskrit while Urdu derives its vocabulary from Persian and Arabic. For their translation, the Hindi word of Sanskrit origin भग्वान [bʰəgvɑn] (God) must be translated into the Urdu word الله [əlləh] (God), a word of Arabic origin, instead of its literal transliteration بھگوان (not an acceptable expression for the Urdu community) and vice versa. Figure 10 shows an example sentence to explain the lexical divergence issue for Hindi–Urdu translation. Lexically translated words are marked in bold.

[14] We will explain the term Indo-Pak languages in the next chapter.

[15] The word is read from right-to-left

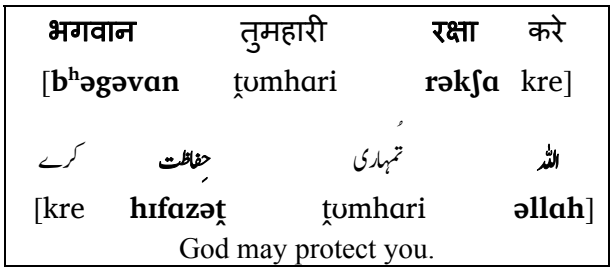


Figure 10: Hindi-Urdu example for dialectal translation

Lexical divergence between different dialects of the same language not only requires a lexical or word for word translation but also requires a partial syntactic analysis of at least elementary phrases like noun phrases, verb phrases, *etc.*, when the source word and translated words have different linguistic properties. Hence, lexical divergence increases the complexity of scriptural translation for *interdialectal translation*.

Lexical divergence is not relevant for pure scriptural translation as we only do transliteration or transcription, but the decision that a word has to be transliterated or translated is crucial for interdialectal translation. It strongly affects user's satisfaction and acceptability of the final output of interdialectal translation.

1.3.3. Challenges and barriers between closely related languages

1.3.3.1. Characteristics

Closely related languages are those languages that are linearly similar to each other. In other words, the word order is mostly the same, but they are still more distant from each other than different dialects of a language. The following problems are important for their translation.

- Script divide
- Absence of necessary information
- Different spelling conventions
- Transliterational or transcriptional ambiguities
- Distinctive sound inventories
- Lexical divergence and translational ambiguities

Other complexities are discussed below.

1.3.3.2. Under-resourced dialect or language pairs

For automatic language and speech processing of a language, basic resources like a raw corpus, a *Part-Of-Speech* (POS) tagged corpus, a morphological analyzer, a parser, *etc.* are crucial. Especially, such resources are necessary inputs to language-independent algorithms and methods in NLP, whether rule-based, empirical or statistical. Only few languages like English, French, German, Arabic, Japanese, Chinese, *etc.* have such computational resources, and the majority of the languages of the world are hanging behind in this aspect of NLP and computational linguistics. The lack of computational resources is a major obstacle for the NLP development of many languages.

In the case of transliteration, all previous methods described in Section 1.2 require large computational resources, *e.g.* pronunciation dictionaries, parallel word lists for training, POS taggers, monolingual or multilingual corpora, *etc.* Thus computational resources play an important role in developing transliteration systems.

Languages and dialects of Indo-Pak languages are under-resourced. In case of the Hindi–Urdu pair, not a single parallel resource existed before 2007. Similarly, the other Indo-Pak languages have very few computational resources, especially Punjabi, Sindhi, Seraiki and Kashmiri. In the very recent past, there existed not a single font for representing the Punjabi and Seraiki languag-

es in the computer and a standard code page does not yet exist for these languages. The standard code page for Urdu was only recently developed, from 1999 to 2001 [2, 79], and was approved by the National Language Authority (NLA) of Pakistan in 2001.

1.4. Approaches for Scriptural Translation

In Section 1.2, we have explored various approaches for general transliteration. Now, we will explore and describe different approaches for solving the scriptural translation problems.

1.4.1. Direct programming approaches

In [119], we used a direct programming and grapheme-based approach for scriptural translation between two mutually incomprehensible scripts of Punjabi (Shahmukhi and Gurmukhi). We gave direct mappings between Shahmukhi and Gurmukhi graphemes and used dependency rules for handling contextual mappings.

We gave an *object-oriented model* for our Shahmukhi to Gurmukhi conversion and implemented the scriptural translation process as a class of Visual Basic .Net with 3,355 lines of code.

We can develop such classes for each step of scriptural translation for various script, dialect or language pairs, but first of all, it is not feasible to develop large classes and secondly, changing a mapping or a contextual rule in the code is not an easy task. To give an example, a part of the code of the Shahmukhi to Gurmukhi transliteration class from [119] is shown in Figure 11. The code implements a simple rule of Shahmukhi to Gurmukhi transliteration saying that a sequence of Alef (ا) plus Zer (ِ) will be converted into ਈ [i] when the sequence is followed by a Choti-Yeh (ی), and otherwise it will be converted into ਇ [ɪ] in Gurmukhi. The study of these mappings and contextual rules show that they can be easily implemented with a *finite-state approach* using a Specialized Language for Linguistic Programming (SLLP) [29, 30, 179]. Finite-state rules equivalent to the code in Figure 11 are given in the XFST [12] in Figure 12.

```
If str.Chars(i + 1) = "ِ" Then 'zer
    If i < str.Length - 2 Then
        If str.Chars(i + 2) = "ی" Then
            If i < str.Length - 3 Then
                If str.Chars(i + 3) = "ا" _
                OrElse str.Chars(i + 3) = "ے" _
                Then
                    convertedString += "ਇ"
                    i += 1
                Else
                    convertedString += "ਈ"
                    i += 2
                End If
            Else
                convertedString += "ਈ"
                i += 2
            End If
        Else
            convertedString += "ਇ"
            i += 1
        End If
    Else
        convertedString += "ਇ"
        i += 1
    End If
```

Figure 11: Direct programming example [125]

```
Read regex [[ا ِ] -> ਇ];
Read regex [[ا ِ] -> ਈ || _ ی];
```

Figure 12: XFST code example

The second line of the XFST code implements the conversion rule taking into account the Choti-Yeh (ی) context. The comparison of the above two code examples clearly shows that it is much easier to handle and change the XFST code than the direct programming code. In addition to ease to construct and ease to manage, the *finite-state approach* also possesses other advantages over the direct programming approach like space and time efficiency.

1.4.2. Finite-state approaches

Finite-state machines (acceptors and transducers) have been used in many areas of NLP and computational linguistics. They provide efficient, versatile and flexible tools for the representation of some linguistic phenomena. *Finite-state machines* are very robust, time and space efficient and can be used to describe easily the local linguistic phenomena encountered in the empirical study of NLP. They lead to a compact representation of dictionaries, lexical rules, idioms and clichés, *etc*. Graphic tools permit to visualize and modify finite-state machines, which helps in correcting and completing lexical rules, grammars, *etc*. [134, 135]

Many researchers [3, 4, 15, 16, 100, 101, 174, 180, 181] have efficiently used *weighted finite-state transducers* for doing transliteration between different pairs of languages like English–Arabic, English–Japanese, *etc*. and also have used different kinds of rich linguistic resources like pronunciation dictionaries. In [121], we used non-probabilistic finite-state transducers for building a Hindi–Urdu scriptural translation system.

All the advantages of *finite-state technology* and its successful use in various fields of NLP make it a strong candidate for the purpose of *translational transliteration*. Finite-State Transducers (FSTs) possess certain very useful and important properties like composition, union, intersection, *etc.* These properties enable us to divide our large and complex problem into small parts that can be solved easily and efficiently. Finally, these small problems can be combined together using the all above properties of FSTs.

1.4.3. Empirical, machine learning and SMT approaches

Recently, statistical, machine learning and Statistical Machine Translation (SMT) techniques have also been employed to solve the problem of transliteration [1, 47, 50, 59, 65, 73, 84, 85, 97, 124, 130, 143, 144, 146, 165, 168, 180, 181] using different methods for transliteration like *grapheme-based*, *phoneme-based*, *etc.*

The construction of transliteration models based on *statistical methods* or SMT is fast and swift, but the availability of resources required for training and development is mandatory. In the case of under-resourced languages, SMT is not a good choice to construct a translation or transliteration system. In the case of scriptural translation, majority script pairs or language pairs are also under-resourced, thus it is difficult to check the suitability of SMT for the construction of translation and transliteration systems for the considered languages.

In the case of Hindi and Urdu, we have a little parallel resource to check the suitability of SMT for scriptural translation of the Hindi–Urdu pair, but such resources are rare for other pairs. We will explore the possibilities of developing parallel resources for SMT systems to translate under-resourced script pairs and language pairs interactively on the Web, using techniques developed for scriptural translation.

1.4.4. Hybrid approaches

We have mentioned in section 1.1.3 *hybrid methods* that use both grapheme and phoneme knowledge for transliterating between languages. Here *hybrid approach* means to combine different transliteration approaches like finite-state, statistical, machine learning, *etc.* together to solve the problem of scriptural translation by exploiting different advantages of different approaches, because it is possible that we may not achieve or produce sufficiently good results that are acceptable, using finite-state or statistical approaches. To improve our results, we will examine and analyze different approaches and study how they can be combined together for transliteration and translational solutions. We introduce a hybrid approach by combining finite-state approaches with statistical approaches to improve and achieve better results for the purpose of scriptural translation.

1.4.5. Interactive approaches

Current MT systems are still far from being perfect after several decades of research in the field. In practice, the output from the current MT systems needs to be post-edited or corrected for errors. We are also not optimistic to achieve 100% accuracy for *scriptural translation*, *inter-dialectal* translation and other *weak translation problems* for closely related languages.

As fully automated high quality transliteration is impossible due to sound inventory differences, different spelling conventions across writing systems, diverse vocabulary and other problems discussed above. The absence of information in one script is also crucial for the correct transliteration into the other script. A scriptural translation model based on weighted graphs (loop-free) is analyzed and explored such that re-ranking can be done interactively and we can achieve high quality results. Thus we will also examine and analyze to introduce interactivity at the end of an automatic transliteration or translation process to achieve good, reasonable, satisfactory, acceptable and usable results in day to day life.

Chapter 2. Scriptural Translation Using FSTs and a Pivot UIT

Finite-state Machines (FSMs) have been successfully used in various domains of Computational Linguistics and Natural Language Processing (NLP). The successful use of FSMs have already been shown in various fields of computational linguistics like lexical analysis, compilation of large scale dictionaries, morphology and phonology, local syntax, syntax, text-to-speech synthesis, speech recognition, Machine Translation and Machine Transliteration [3, 4, 15, 16, 87, 89, 90, 98-101, 131-135, 161, 162, 166, 167, 174, 180, 181]. All these practical and advantageous features of FSMs make them very strong candidates to be used for solving the *scriptural translation problems*.

There are several platforms for developing FSMs, like the Finite-State Machine library (FSM library[16]) by AT&T, WFST: (Finite-state Template library in C++[17] 2000), Xerox Finite-State Technology (XFST)[18] [12], FSA [86] and OpenFST[19], a very recent and open source beta finite-state library based on the FSM library by AT&T. Good and well-established candidates for developing *finite-state models* for scriptural translation problems are FSM Library by AT&T, XFST by Xerox and OpenFST by Google Research and New York University.

We need a platform that can handle Unicode[20] and its UTF8[21] encoding as we have to handle multilingual texts and a range of different alphabets of various languages. In 2006, the simple *text-file* coding format of AT&T FSM Library could not handle Unicode characters according to the best knowledge of the author. OpenFST was not available at the time when we started our work and it is very recent and still needs time to mature.

Thus our only choice to develop our finite-state models was XFST. Indeed, (a) XFST supports Unicode and UTF8, (b) XFST is easy to use and is described in a very simple and comprehensive way by [12] and (c) its coding conventions are very easy. In particular, one can easily code the contextual dependencies as shown in Figure 12 of the previous chapter. All these characteristics make XFST a very good candidate to build finite-state models.

In this chapter, we investigate the effectiveness and efficiency of Finite-State Transducers (FSTs) for the scriptural translation problems, described in the previous chapter. We also introduce, define and explain a multipurpose *translational pivot* called Universal Intermediate Transcription (UIT). First, we analyze Indo-Pak languages for the *Scriptural Translation problems*. We also describe basic principles and advantages of UIT and develop UIT mappings for Indo-Pak languages. Then, we explain and describe *finite-state scriptural translation models*. Finally we describe our experimental setup and discuss our results.

[16] http://www.research.att.com/~fsmtools/fsm

[17] http://membres.lycos.fr/adant/tfe

[18] http://www.stanford.edu/~laurik/fsmbook/home.html

[19] http://www.openfst.org

[20] http://www.unicode.org

[21] http://www.utf-8.com

2.1. Scriptural Translation for Indo-Pak Languages

The Indian subcontinent is a linguistically rich area. India alone represents 438 languages [114] belonging to several major language families, the two largest being the Indo-Aryan and the Dravidian families. Pakistan also represents six major languages and 58 minor ones [156]. Punjabi, Sindhi, Seraiki and Kashmiri exist on both sides of the common border between India and Pakistan and all of them are written in two or more mutually incomprehensible scripts. Hindi, one of the official languages of India, and Urdu, the national language of Pakistan and one of the official languages of India, are considered as two varieties of the same language called Hindustani by [152].

In the words of [157], "One man's Hindi is another man's Urdu." Despite the linguistic facts and probably because languages are felt to be "the flags of national identities" [64], the Hindi and Urdu communities claim that Hindi and Urdu are two different languages [155-157]. Punjabi and Seraiki also have this kind of controversy [155, 156]. The Hindi–Urdu pair exists both in India and Pakistan or rather we should say that Urdu exists in both countries. Like the bordering languages, Hindi is written in Devanagari script and Urdu is written in a derivation of the Persio-Arabic script. We call all these languages the *Indo-Pak* languages because they exist both in India and Pakistan. A family tree for the Indo-Pak and other major Pakistani languages is given in Figure 13.

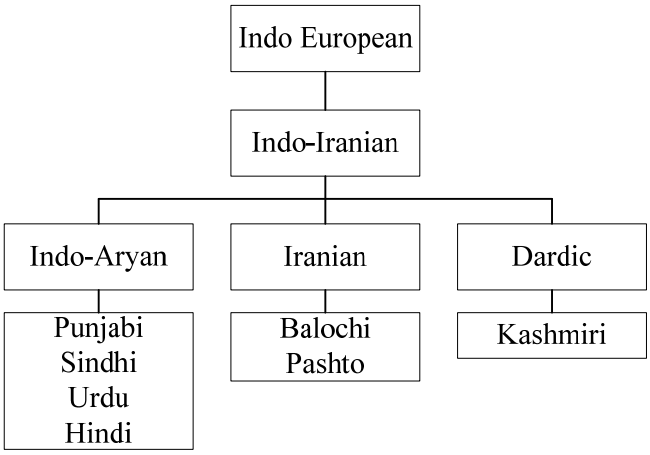


Figure 13: Family tree of the Indo-Pak and other Pakistani languages

In contrast to the scriptural dissimilarities, different varieties of the Indo-Pak languages share grammar, morphology, the major vocabulary, history, classical literature and cultural heritage. *Scriptural translation* of the Indo-Pak languages is an attempt to bridge the scriptural, ethnical, cultural and geographical divisions among the Indo-Pak communities around the world.

2.1.1. Scripts of Indo-Pak languages

The scripts of the Indo-Pak languages can be divided into two categories (1) scripts based on the Persio-Arabic script, and (2) Indic scripts.

2.1.1.1. Scripts based on the Persio-Arabic script

Pakistani varieties of all Indo-Pak languages derive their alphabets from the Persio-Arabic alphabet. Only Urdu and Kashmiri are written both in India and Pakistan in the derived Persio-Arabic script. They are also written in Devanagari in India alone. All of them except Sindhi are traditionally written in the Nasta'leeq writing style, a highly cursive, context-sensitive, beautiful and calligraphic style for languages written in the Arabic script or in its derivations [120]. Sindhi is traditionally written in Naskh style like the Arabic and Persian languages, but it can also be written in the Nasta'leeq style [120].

The distinguishing characteristics of scripts based on the Persio-Arabic script are presented now for the benefit of the unacquainted reader. They are read from *right-to-left*. Unlike English, characters do not have upper and lower cases. The shape assumed by a character in a word is context-sensitive, i.e. the shape is different depending upon whether the position of the character is at the beginning, in the middle or at the end of the constituent word. This generates three shapes, the fourth being the independent shape of the character [117-119, 121, 122, 194]. Figure 14 shows these four shapes of the character BEH (ب) in the Naskh writing style.

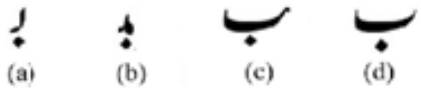


Figure 14: Context sensitive shapes of Beh [194]

To be precise, the above is true for all except certain characters that only have the independent and the terminating shape when they come at the beginning and in the middle or at the end of a word respectively, *e.g.* Alef (ا), Dal (د), Reh (ر), *etc.* [117, 118, 122, 194]. Arabic, Persian and Pakistani languages have a large set of diacritical marks that are necessary for the correct articulation of a word. The diacritical marks appear above or below a character to define a vowel, to build a compound word, or to geminate a character [117-119, 121, 122, 194]. They are the foundation of the vowel system in these languages. The most common diacritical marks (fatha, kasra and dumma) with the character BEH are shown in Figure 15.

Figure 15: BEH with common diacritics

Diacritics are part of the script, but are sparingly used. They are essential for ambiguities removal, NLP, and speech synthesis [78, 118-122, 194, 195]. More detailed information about Indo-Pak scripts based on the Persio-Arabic script is given in Annex 1.

2.1.1.2. Indic scripts

Two Indic scripts, Devanagari and Gurmukhi, are used for the Indo-Pak languages. Devanagari is used for all Indo-Pak languages and Gurmukhi is used only for Punjabi and Seraiki in India. Both Indic scripts are read from *left-to-right* and are partially syllabic because every consonant inherits the vowel [ə] [14, 94, 118, 119, 121, 136, 152]. For example, Devanagari consonants क [k], ब [b] and स [s] represent the [kə], [bə] and [sə] sounds respectively. In Gurmukhi, the consonants ਕ [k], ਬ [b] and ਸ [s] also represent the [kə], [bə] and [sə] sounds respectively. Both Devanagari and Gurmukhi have vowel signs to mark other vowels with consonants [14, 94, 118, 119, 121, 136, 152]. For example, [ko], [kɑ], [kɪ] and [ki] are represented by क + ो = को, क + ा = का, क + ि = कि and क + ी = की in Devanagari and by ਵ + ੋ = ਵੋ, ਵ + ਾ = ਵਾ, ਵ + ਿ = ਵਿ and ਵ + ੀ = ਵੀ in Gurmukhi respectively.

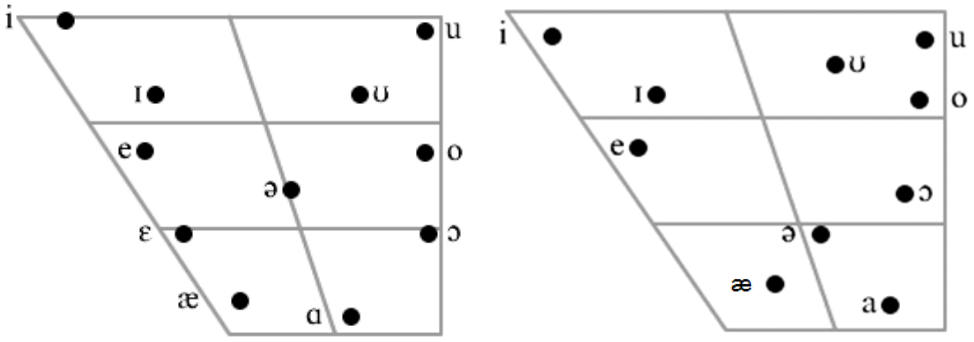
In Devanagari, the VIRAMA (Halant) sign (्) is used to mark the absence of the inherited vowel [ə] between two consonants. Two or more consonants can be combined together to form a cluster called Conjunct by marking the absence of the inherited vowel between two consonants [94, 118, 121, 136]. Conjuncts are contextual shapes of Devanagari in which the original shape of a consonant changes. For example, in the word हिन्दी [hɪnd̪i] (Hindi), there is a conjunct form न्द [nd̪] of consonants न [n] and द [d̪] and the consonant न [n] changes its shape. More detailed information on Devanagari and Gurmukhi scripts is given in Annexes 2, 3 and 4.

2.1.2. Analysis of Indo-Pak scripts for scriptural translation

Characters of the Indo-Pak languages can be divided into various groups according to their types like vowels, consonants, diacritics, digits, punctuation marks and other symbols.

2.1.2.1. Vowel analysis

A vowel is a sound that is produced with an open vocal tract so that there is no build-up of air pressure at any point above the glottis[22]. Hindi and Kashmiri have 11 and 17 vowels respectively. The rest of the Indo-Pak languages have 10 vowels that are common to all these languages. Figure 16 shows vowel charts for Hindi, Urdu and Sindhi. The vowel [ɛ] is shown in the chart of Hindi–Urdu vowels, but it has no defined representation or character in both Hindi and Urdu and is represented by the vowel [ə] [78]. The Hindi–Urdu vowel chart does not show the vocalic RA vowel [r̥] of Hindi.



Hindi–Urdu Vowel Chart **Sindhi Vowel Chart**[23]

Figure 16: Hindi, Urdu and Sindhi vowel chart (source Wikipedia)

Devanagari has in total 11 independent vowel characters, *e.g.* इ [ɪ], औ [ɔ], *etc.* and 10 dependent vowel signs (also called Matras), *e.g.* ि [ɪ], ौ [ɔ], *etc.* to represent vowels of Indo-Pak languages. Kashmiri has 17 vowels and it is not possible to render them with the available 11 vowel characters and 10 vowel signs of Devanagari, so we have to make some approximations for doing transliteration from Kashmiri's derived Persio-Arabic script to Devanagari. These approximations make Kashmiri vowel conversion from Devanagari a really hard and complex task. More details are given in Annexes 2 to 4.

The vowel representation in scripts based on the Persio-Arabic script is highly complex and context-sensitive [78, 118-121]. The vowels are represented with the help of five vowel characters ALEF (ا), ALEF-MADA (آ), WAW (و), YEH (ی), BARI-YEH (ے) and AIN (ع) and 20 diacritical marks. Urdu, Punjabi/Shahmukhi, Sindhi/Sindhi and Seraiki/Shahmukhi possess the same set of 15 diacritical marks, *e.g.* ZABAR (َ), ZER (ِ), PESH (ُ), SHADDA (ّ), *etc.* Kashmiri adds 5 diacritical marks and 4 characters to represent its additional vowels that are shown in Table 10. More detailed information on scripts based on the Persio-Arabic script is given in Annex 1.

Additional Kashmiri Vowel Characters	
Diacritics	ٔ [əː] ٕ [ɨː] ٚ [oː] ٔ [ə] ٕ [ɨ]
Characters	و [ɔ] ۆ [oː] ێ [e] ی [j]

Table 10: Additional Kashmiri vowel characters and diacritics

A detailed analysis of 10 common vowels of Indo-Pak languages for Urdu and Punjabi is given in Table 11.

[22] http://en.wikipedia.org/wiki/Vowel

[23] Sindhi vowel chart is modified and corrected for an error in it.

Vowel	Contextual nature of vowels in scripts based on the Persio-Arabic script
ə	It is represented by ALEF (ا) + ZABAR (ً) at the start of a word e.g. اَب [əb] (now) and by ZABAR (ً) in the middle of a word respectively e.g. رَبّ [rəbb] (God). It never comes at the end of a word.
ɑ	It is represented by ALEF-MADDA (آ) at the start of a word e.g. آدمی [ɑd̪mi] (man) and by ALEF (ا) or ALEF-MADDA (آ) in the middle of a word e.g. جانا [ʤɑnɑ] (go), بلآخر [bɪlɑxər] (at last). At the end of a word, it is represented by ALEF (ا). In some Arabic loan words, it is represented by YEH (ی) + KHARI-ZABAR (ٰ) at the end of a word e.g. اعلیٰ [ə?lɑ] (Superior) and by KHARI-ZABAR (ٰ) in the middle of a word e.g. الٰہی [ɪlɑhi] (God).
e	It is represented by ALEF (ا) + YEH (ی) at the start of a word e.g. ایثار [esɑr] (sacrifice), and by YEH (ی) or BARI-YEH (ے) in the middle of a word e.g. میرا [merɑ] (mine), بےگھر [begʰər] (homeless) etc. At the end of a word, it is represented by BARI-YEH (ے) e.g. سارے [sɑre] (all).
æ	It is represented by ALEF (ا) + ZABAR (ً) + YEH (ی) at the start of a word e.g. ایہہ [æh] (this) and by ZABAR (ً) + YEH (ی) in the middle of a word e.g. مَیل [mæl] (dirt). At the end of a word, it is represented by ZABAR (ً) + BARI-YEH (ے) e.g. ہَے [hæ] (is).
ɪ	It is represented by ALEF (ا) + ZER (ِ) at the start of a word e.g. اِس [ɪs] (this) and by ZER (ِ) in the middle of a word e.g. بارش [bɑrɪʃ] (rain). It never comes at the end of a word. At the end of a word, it is used as KASR-E-IZAFAT to build compound words.
i	It is represented by ALEF (ا) + ZER (ِ) + YEH (ی) at the start of a word e.g. اِیمان [imɑn] (belief) and by ZER (ِ) + YEH (ی) in the middle or at the end of a word e.g. امیری [ɑmiri] (richness), قریب [qərib] (near), etc.
ʊ	It is represented by ALEF (ا) + PESH (ُ) at the start of a word e.g. اُدّھر [ʊd̪d̪ʰər] (there) and by PESH (ُ) in the middle of a word e.g. مُلّ [mʊll] (price). It never comes at the end of a word.
u	It is represented by ALEF (ا) + PESH (ُ) + WAW (و) at the start of a word e.g. اُونگھتا [ũgʰəʈɑ] (dozzing) and by PESH (ُ)+ WAW (و) in the middle or at the end of a word e.g. صُورت [surəʈ] (face), تراز‌ُو [ʈərɑzu] (physical balance), etc.
o	It is represented by ALEF (ا) + WAW (و) at the start of a word e.g. اوچھا [oʧʰɑ] (nasty) and by WAW (و) in the middle or at the end of a word e.g. ہولی [holi] (slowly), کہو [kəho] (say), etc.
ɔ	It is represented by ALEF (ا) + ZABAR (ً) + WAW (و) at the start of a word e.g. اَوٹ [ɔʈ] (hindrance) and by ZABAR (ً) + WAW (و) in the middle or at the end of a word e.g. مَوت [mɔʈ] (death).

Table 11: Vowels analysis for Urdu and Punjabi in scripts based on the Persio-Arabic script [119, 121]

2.1.2.2. Consonant analysis

Consonants of the Indo-Pak languages can be further divided into (1) aspirated consonants and (2) non-aspirated consonants.

2.1.2.2.1. Aspirated consonants

Unlike in Arabic and Persian, the phenomenon of aspiration exists in all Indo-Pak languages. Hindi and Urdu have 15 aspirated consonants. In Urdu, Punjabi/Shahmukhi, Seraiki/Shahmukhi and Kashmir, a special character HEH-DOACHASHMEE (ھ) is used to mark the aspiration of a consonant and aspirated consonant is represented by a combination of a consonant to be aspirated and HEH-DOACHASHMEE (ھ), *e.g.* ب [b] + ھ [h] = بھ [bʰ], ج [ʤ] + ھ [h] = جھ [ʤʰ], ر [r] +

ہ [h] = ہر [r^h], *etc.* [118, 119, 121, 152]. On the other hand, Sindhi employs two different ways to mark the aspiration of a consonant: either it introduces a new character for the aspirated version of a consonant or it uses HEH-DOACHASHMEE (ھ) like other languages [122]. Numbers of aspirated consonants for each Indo-Pak language are shown in Table 12.

Aspirated Consonants of Indo-Pak Languages	Hindi	Urdu	Punjabi	Sindhi	Seraiki	Kashmiri
	15	15	16	11	16	6

Table 12: Numbers of aspirated consonants in Indo-Pak languages

Devanagari and Gurmukhi also employ two different ways to represent aspirated consonants. Devanagari has 11 separate characters for aspirated consonants, *e.g.* भ [b^h], झ [$ʤ^h$], *etc.* For the rest of aspirated consonants, it uses a conjunct form with HA (ह), *e.g.* र [r] + ् + ह [h] = र्ह [r^h] [94, 118, 120, 121, 136]. More details are given in Annexes 1 to 4.

2.1.2.2.2. Non-aspirated consonants

Urdu, Punjabi/Shahmukhi, Seraiki/Shahmukhi, Sindhi and Kashmiri (Urdu script) have added new characters for indigenous sounds like retroflex and aspirations [118, 119, 121, 122].

Although native speakers of the Indian subcontinent cannot distinguish between the Arabic sounds SHE (ث) [ɵ], SEEN (س) [s] and SAD (ص) [$s^ʕ$] and produce a sound [s] for these characters, all derived Persio-Arabic scripts for Indo-Pak languages adopted these characters. All these cases are shown in Table 13.

Sound	[ʈ]	[s]	[z]	[h]
Characters	ت، ط، ۃ	ث، س، ص	ذ، ز، ژ، ض، ظ	ح، ھ، ہ
Devanagari / Gurmukhi	त / ਤ	स / ਸ	ज़ / ਜ਼	ह / ਹ

Table 13: One native sound for multiple characters

Urdu and Kashmiri possess in total 35 non-aspirated consonant characters to represent 25 sounds and Punjabi has 36 consonant characters to represent 26 sounds. On the other hand, Sindhi and Seraiki have 40 and 41 consonant characters to represent 30 and 32 sounds respectively.

In contrast to the derived Persio-Arabic scripts, Devanagari and Gurmukhi are descendents of the indigenous scripts of the region like Brahmi, Sharda, *etc*. [14, 94] and they follow the doctrine of one character for one sound. Thus, both Devanagari and Gurmukhi have only one character for each case given in Table 13. More details are given in Annexes 1 to 4.

2.1.2.3. Diacritics analysis

Diacritical marks are the backbone of the vowel systems in Urdu, Punjabi/Shahmukhi, Sindhi, Seraiki/Shahmukhi and Kashmiri (Urdu script). They are detailed in Table 11. Consider the example of Figure 17 for Urdu. The circled words in the sentences of Figure 17(a) and Figure 17(b) are exactly the same in the written form. But while looking at the IPA transcription of the words right blow them, we observe that the word has two different pronunciations and consequently two different meanings in these sentences. They are pronounced [ʧɔɽi] (wide) and [ʧuɽi] (bangle) in Figure 17(a) and Figure 17(b) respectively. In phonetic transcriptions, the vowel after the first consonant of these words is different because the diacritical marks ZABAR (ً) and PESH (ُ) are missing. These words with diacritics are also shown in the Figure 17.

A human can read and understand each sentence due to his tuned mind that knows how to read text without diacritical marks. But in the absence of diacritical marks, it becomes difficult to understand the phonetics and meanings of a written text for automatics language processing. More details are given in Annex 1.

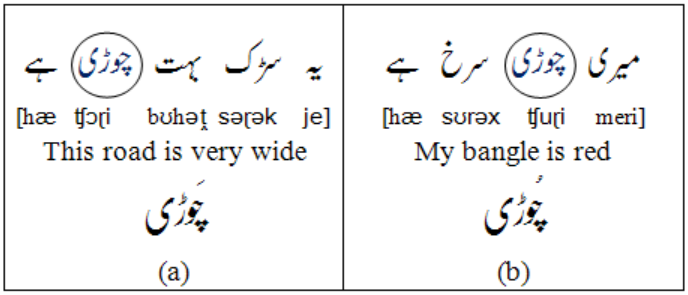


Figure 17: Ambiguity due to missing diacritical marks

2.1.2.4. Punctuations, digits and special characters analysis

Urdu, Punjabi/Shahmukhi, Sindhi, Seraiki/Shahmukhi and Kashmiri use the Arabic punctuation marks and also possess a set of special symbols that come from various sources like religion, poetry, *etc.* A few examples of these special symbols are ﷺ, ؐ and ؒ. These special symbols have no equivalent in Devanagari and Gurmukhi.

Only some of them must be treated when we perform scriptural translation because they mean full words or phrases, like the ALLAH ligature (ﷲ), the MUHAMMAD ligature (ﷺ) and the BISMILLAH ligature (بسم الله الرحمن الرحيم).The rest of these special symbols are ignored during the translation process. All these languages have 10 digits and a decimal sign. Detailed information on punctuations and special symbols is given in Annexes 1 to 4.

2.1.3. Indo-Pak scriptural translation problems

We have already discussed various challenges and barriers of scriptural translation and illustrated them with several examples from the Indo-Pak languages in the previous chapter.

We just list them here to refresh the memory: *scriptural divide*, *under-resourced and under-written characters of the languages*, *absence of necessary information*, *differences in spelling conventions*, *transliterational or transcriptional ambiguities*, *differences in sound inventories*, *lexical divergence, translational ambiguities* and *under-resourced character of dialect or language pairs*.

2.2. Universal Intermediate Transcription (UIT)

General translation opens a way of communication between the users of two different languages. Scriptural translation also provides a communication means between the two communities that use the same language and can communicate verbally, but not scripturally. They become alien to each other like their scripts. That is illustrated in Figure 6 with a Hindi–Urdu example. Figure 18 shows another example the Punjabi/Shahmukhi–Punjabi/Gurmukhi pair.

پنجابی ساڈی ماݨ جوگی ماں بولی اے

ਪੰਜਾਬੀ ਸਾਡੀ ਮਾਣ ਜੋਗੀ ਮਾਂ ਬੋਲੀ ਏ

[pɲʤabi saɖi maɳ ʤogi maɲ boli e]

Punjabi is our respectable mother tongue.

Figure 18: Example of Punjabi/Shahmukhi - Punjabi/Gurmukhi pair

Therefore, we were considering the problem of scriptural translation as a problem of translation. Like in general translation, we introduce the concept of pivot for scriptural translation and call it Universal Intermediate Transcription (UIT). It will not only serve the role of Pivot for the Lan-

guage/Script pairs of the same languages written in different scripts, but also play a role of pivot for general translation across the Indo-Pak language pairs. For example, UIT could also be used as a pivot for the Urdu to Punjabi translation and vice versa.

The Indo-Pak languages are very close to each other and linearly very similar. A pure scriptural translation can serve as a translation between different pairs of the Indo-Pak languages because people can read and even understand largely the meaning of a scripturally translated text (translated into the script of their usage).

We can improve significantly this rough translation by adding a *word-to-word translation* process. Scriptural translation between the Indo-Pak languages using a pivot is a first step towards a fully working *Indo-Pak machine translation system* and the development of *collaborative parallel framework* like ParGram[24] [35, 36, 38, 39, 96] to develop multilingual linguistic resources.

An example sentence "this is my house" of Hindi–Urdu–Punjabi (Shahmukhi and Gurmukhi) is shown in Table 14. Hindi, Urdu and Punjabi words are aligned in columns. Words that are subject to *word for word translation* are given in bold (first and last words).

Hindi	**ये** je	मेरा merɑ	घर g^hr	**है** hæ
Urdu	**یہ** je	میرا merɑ	گھر g^hr	**ہے** hæ
Punjabi/Shahmukhi	**ایہ** æh	میرا merɑ	گھر g^hr	**اے** e
Punjabi/Gurmukhi	**ਐਹ** æh	ਮੇਰਾ merɑ	ਘਰ g^hr	**ਏ** e
	this	my	house	**is**

Table 14: Example of scriptural translation between Indo-Pak languages

2.2.1. What is UIT?

UIT is a multipurpose pivot. In the current study, it is used as a *phonetico-morphotactic* pivot for the *surface morphotactic translation* or scriptural translation, as a *phonetico-morphological* pivot for *word for word translation* (lexical translation) and as a *phonetico-morphosyntactic* lexical pivot for *syntax-based translation* (in conjunction with syntactic transfer).

In future, this multipurpose pivot will provide the basis for the development of a *collaborative parallel framework* for Indo-Pak languages, in which we can develop multilingual morphological analyzers and syntactic grammars for the Indo-Pak languages because they share a large part of their complex phenomena of morphology, syntax, *etc*.

The Indo-Pak languages possess a common *phonetic repertoire* with a few additions or subtraction for each language or dialect. The common phonetic repertoire serves as the central *phonetic lexicon* for the UIT encoding of these languages. In other words, UIT is the phonetic encoding of a language such that a text written in the writing system of a language can be converted de-

[24] A parallel grammar development project in the framework of Lexical Functional Grammar for multiple languages.

terministically into UIT encoded text and the original text can be regenerated from the UIT encoded text without any ambiguity.

2.2.2. Principles of UIT

For each group G of languages and dialects that are very similar to each other, we propose to define a pivot representation what we call "UIT-G". We have built a "UIT-Indo-Pak" pivot that we call UIT for short in the sequel.

UIT is a phonetic encoded pivot. IPA is a good choice for the UIT encoding scheme because the IPA associated with each character of the Indo-Pak languages serves as the encoding standard. But, portability and readability of IPA characters were not very good across different computer systems and operating systems in the recent past, when we selected a good candidate for the UIT encoding scheme in 2006.

Although we have not used IPA as encoding scheme, we have used the IPA coding associated with each character as the encoding principle for our ASCII encoding scheme. We selected the printable ASCII characters to base the UIT encoding scheme because it is universally portable to all computer systems and operating systems without any problem [31, 69, 184]. UIT is a deterministic and unambiguous scheme of transcription for Indo-Pak languages in ASCII range 32–126, since a text in this rage is portable across computers and operating systems [69, 184], using the associated IPA with a character as an encoding principle.

Speech Assessment Methods Phonetic Alphabet (SAMPA)[25] is a widely accepted scheme for encoding IPA into ASCII. The purpose of SAMPA was to form the basis of an international standard machine-readable phonetic alphabet for the purpose of international collaboration in speech research [184]. The UIT encoding of Indo-Pak languages is developed as an extension of the SAMPA and X-SAMPA that covers all symbols on the IPA chart [184].

2.2.3. UIT mappings for Indo-Pak languages

The sound or IPA associated with each character of the Indo-Pak languages is used as the main part of the encoding of a character in UIT. All characters of the Indo-Pak languages are subdivided into three categories, consonants, vowels and other symbols (punctuations and digits).

Consonants are further divided into aspirated consonants and non-aspirated consonants. For aspiration, in phonetic transcription a simple 'h' following the base consonant symbol is considered adequate [184]. In the Indo-Pak languages, we have two characters with IPA [h]. Thus to distinguish between the 'h' consonants and the aspiration, we use *underscore* '_' to mark the aspirate and we encode an aspiration as '_h'. For example, the aspirated consonants ٹھ [$ʈ^h$], پھ [p^h] and ڇھ [$ʧ^h$] of the Indo-Pak languages are encoded as 't`_h', 'p_h' and 't_S_h' respectively. Similarly for the dental consonants, we use the '_d' marker. For example, the characters ڈ [$d̪$] and ٹ [$t̪$] are encoded as 'd_d' and 't_d' in UIT. Table 15 shows the UIT encodings of aspirated consonants of the Indo-Pak languages.

The IPA symbols for retroflex consonants of the Indo-Pak languages are ʈ, ɖ, ɽ, ɭ and ɳ. SAMPA uses ASCII-96 to encode the retroflex consonants [184]. Therefore the UIT encodings of the IPA characters ʈ, ɖ, ɽ, ɭ and ɳ are 't`', 'd`', 'r`', 'l`' and 'n`' respectively.

Table 13 above in section 2.1.2.2.2 shows cases where several characters are used for the same sound in the Indo-Pak languages. We distinguish these characters in the UIT encoding by adding a number (1, 2,…) after the UIT encoding of the IPA associated with their sound. For example, the Urdu, Punjabi/Shahmukhi, Seraiki/Shahmukhi, Sindhi and Kashmiri characters ث, س

[25] http://www.phon.ucl.ac.uk/home/sampa/

and ص are encoded into UIT as ‘s1’, ‘s’ and ‘s2’ respectively. The encoding of س is ‘s’ because it occurs more frequently than the other characters. Tables of UIT encodings of all other consonants, vowels and other symbols of the Indo-Pak languages are given in Annex 5.

IPA	Hindi	Gurmukhi	Kashmiri	Punjabi	Seraiki	Sindhi	Urdu	UIT
b^h	भ	ਭ		بھ	بھ	ڀ	بھ	b_h
p^h	फ	ਫ	پھ	پھ	پھ	ڦ	پھ	p_h
$ʈ^h$	थ	ਥ	تھ	تھ	تھ	ٿ	تھ	t_d_h
$ʈ^h$	ठ	ਠ	ٹھ	ٹھ	ٹھ	ٺ	ٹھ	t`_h
$ʤ^h$	झ	ਝ		جھ	جھ	جھ	جھ	d_Z_h
$ʧ^h$	छ	ਛ	چھ	چھ	چھ	ڇ	چھ	t_S_h
$ɖ^h$	ध	ਧ		دھ	دھ	ڌ	دھ	d_d_h
$ɖ^h$	ढ	ਢ		ڈھ	ڈھ	ڍ	ڈھ	d`_h
r^h	र्ह	ਰ੍ਹ		رھ	رھ	ڙھ	رھ	r_h
$ɽ^h$	ढ़	ੜ੍ਹ		ڑھ	ڑھ		ڑھ	r`_h
k^h	ख	ਖ	کھ	کھ	کھ	ک	کھ	k_h
g^h	घ	ਘ		گھ	گھ	گھ	گھ	g_h
l^h	ल्ह	ਲ੍ਹ		لھ	لھ		لھ	l_h
m^h	म्ह	ਮ੍ਹ		مھ	مھ		مھ	m_h
n^h	न्ह	ਨ੍ਹ		نھ	نھ		نھ	n_h
v^h		ਵ੍ਹ		وھ	وھ			v_h

Table 15: UIT encoding of aspirated consonants of the Indo-Pak languages

2.3. Finite-state Scriptural Translation model

Finite-state technology has been successfully used in various domains of NLP [134, 135, 162]. Following [4, 100, 101, 131, 134, 135, 162, 174], we use Finite-State Transducers (FSTs) for solving the problem of scriptural translation. A *finite-state scriptural translation model* is a combination of serially composed transducers. This model possesses two variants, non-deterministic and deterministic. In this section, we describe non-deterministic and deterministic transducers for scriptural translation of the Indo-Pak languages. Finally we discuss the system architecture of finite-state scriptural translation model.

2.3.1. Finite-state transducers

Both conversions of the source language text into the UIT encoded text and from the UIT encoded text into the target language text are regular relations on strings. Moreover, regular relations are closed under serial composition and a finite set of conversion relations when applied to each other’s output in a specific order, also defines a regular expression [87]. Thus we model the conversions from the source language to UIT and from UIT to the target language as finite-state transducers. We divide both conversion problems into smaller problems and write conversion rules for each reduced problem. At the end, by the composition of these small conversion rules in an ordered sequence, we construct a single transducer that simulates the whole set of rules. This transducer can be integrated in the finite-state computational model for scriptural translation. We have developed such scriptural translation transducers for all Indo-Pak languages. These translational transducers can be deterministic and non-deterministic.

2.3.1.1. Non-deterministic transducers

In the definition of UIT, we said that the correspondences between an Indo-Pak language and UIT and between UIT and a given written form are functional (unambiguous). It is true when we consider a written form containing all information sufficient to uniquely determine the pronunciation of the utterances. For example, the correspondence between a text in Hindi and its UIT encoding is one-to-one.

Non-determinism in the automata appears when the correspondence is not one-to-one, and when we consider the conversion process, because it is often the case that there is not enough information in the character string itself to determine the proper pronunciation of a sentence, even if it is unique. To remove all potential ambiguities caused by the absence of diacritics (Urdu) or by the absence of indication of the absence of a vowel or the presence of the default vowel 'ə' (Hindi), for example, one often needs to do a full syntactic, semantic and in some cases pragmatic analysis of the sentence in discourse (or dialogue).

By non-determinism, we thus mean that ambiguities can appear when we convert a text written in a language L_i in a script W_j into a UIT text and the UIT text is converted into another language L_k in a script W_l. For example, the conversion of the Hindi word सितारा [sɪtɑrɑ] (star) into UIT gives 'sIt_dA1rA1' (deterministic output) and the conversion of 'sIt_dA1rA1' into Punjabi/Shahmukhi gives us a set of 18 possible transcriptions, out of which only one is correct ستارہ. The whole process is shown in Figure 19.

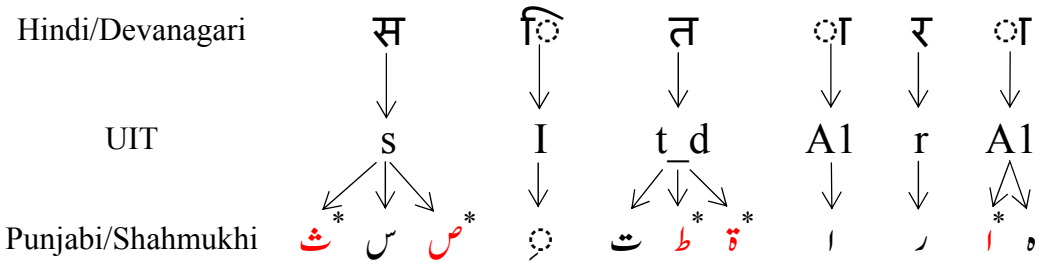


Figure 19: Ambiguities of Hindi/Devanagari to Punjabi/Shahmukhi scriptural translation

There are two types of regular relations that we model with finite-state transducers. The first comprises the *one-to-one mappings* between characters of an Indo-Pak languages and UIT symbols. We call them *character mappings*. The sound or IPA associated with a character serves as the mapping relation between the character and a symbol from the UIT encoding. The others are the *contextual mappings*.

For the discussion on character mappings and contextual mappings, we will consider the example of Figure 19 and the UIT encodings of Table 15 as a small domain of application. We will formulate regular relations for these examples and develop example transducers for this small domain.

2.3.1.1.1. Character mappings

Table 16 gives the UIT encodings of aspirated consonants of Hindi and Punjabi/Shahmukhi, already shown in Table 15, together with their conversion into character mapping regular relations.

IPA	Hindi to UIT Regular Relations	Punjabi/Shahmukhi to UIT Regular Relations
b^h	भ → b_h	ﺑﮭ → b_h
p^h	फ → p_h	ﭘﮭ → p_h
$t̪^h$	थ → t_d_h	ﺗﮭ → t_d_h
$ʈ^h$	ठ → t`_h	ﭨﮭ → t`_h
$ɖʒ^h$	झ → d_Z_h	ﺟﮭ → d_Z_h
$tʃ^h$	छ → t_S_h	ﭼﮭ → t_S_h
$d̪^h$	ध → d_d_h	ﺩﮬ → d_d_h
$ɖ^h$	ढ → d`_h	ﮈﮬ → d`_h
r^h	ऱ्ह → r_h	ﺭﮬ → r_h
$ɽ^h$	ढ़ → r`_h	ﮍﮬ → r`_h
k^h	ख → k_h	ﮐﮭ → k_h
g^h	घ → g_h	ﮔﮭ → g_h
l^h	ल्ह → l_h	ﻟﮭ → l_h
m^h	म्ह → m_h	ﻣﮭ → m_h
n^h	न्ह → n_h	ﻧﮭ → n_h
v^h		ﻭﮬ → v_h

Table 16: Regular rules for aspirated consonants of Hindi and Punjabi/Shahmukhi

By interchanging the UIT encodings before the arrow sign and the respective characters of Hindi and Punjabi/Shahmukhi after the arrow, we can construct regular conversion relations from UIT to Hindi and Punjabi/Shahmukhi. Similarly, regular conversion relations for the characters of the example word of Figure 19 are shown in Table 17.

IPA	Hindi/Devanagari Regular Relations	Punjabi/Shahmukhi Regular Relations
s	स → s	س → s or ث → s1 or ص → s2
ɪ	ि → I	ِ → I
t̪	त → t_d	ت → t_d or ط → t_d1 or ۃ → t_d2
ɑ	ा → A1	ا → A1
r	र → r	ر → r
h	ह → h	ہ → h

Table 17: Regular relations for the example of Figure 19

We can transform these conversion relations into XFST rules. Table 18 gives the sample XFST code for Hindi to UIT scriptural translation corresponding only to the conversion relations of Table 16 and Table 17. The first command (first line) clears the XFST stack of any previous finite-state automaton or transducer, if any. The second command sets the character encoding to UTF8. The third and fourth commands are compiled into different transducers and saved onto the XFST stack when this code is compiled with the XFST compiler. The last command composes all finite-state transducers constructed in the stack into a single finite-state transducer and saves it on the stack. Each 'read regex' (read regular expression) command can consist of one or multiple conversion rules and is terminated by a semicolon. We can then apply the final finite-state transducer to some Hindi text to convert it into UIT. The example is shown in Figure 19.

```
clear stack
set char-encoding UTF-8
read regex [ि -> I];
read regex [ख -> [k "_" h], घ -> [g "_" h], छ -> [t "_" S
  "_" h], झ -> [d "_" Z "_" h], ठ -> [t "`" "_" h], ढ -> [d
  "`" "_" h], थ -> [t "_" d "_" h], ध -> [d "_" d "_" h], फ
  -> [p "_" h], भ -> [b "_" h], ढ़ -> [r "`" "_" h], स -> s,
  त -> [t "_" d], र -> r, ल -> l, म -> m, न -> n, व -> v, ह
  -> h];
read regex [[ृ ह] -> ["_" h] || [र | ल | म | न] _ ];
compose net
```

Table 18: Sample XFST code for Hindi to UIT scriptural translation

Similarly, we can construct the XFST rules for UIT to Punjabi/Shahmukhi scriptural translation, shown in Table 19. The third command defines a constant containing a finite-state automaton that we can use later in constructing transducers, especially for defining the context.

```
clear stack
set char-encoding UTF-8
define CONSONANTS [b | [b "_" h] | p | [p "_" h] | [t "_" d]
| [t "_" d "_" h] | [t "`"] | [t "`" "_" h] | [s 1] | [d "_"
Z] | [d "_" Z "_" h] | [t "_" S] | [t "_" S "_" h] | [d "_"
d] | [d "_" d "_" h] | [d "`"] | [d "`" "_" h] | r | [r "_"
h] | [r "`"] | [r "`" "_" h] | s | [s 2] | [t "_" d 1] | k |
[k "_" h] | g | [g "_" h] | l | [l "_" h] | m | [m "_" h] |
n | [n "_" h] | v | [v "_" h] | h | [t "_" d 2]];
read regex [k -> ک, g -> گ, [t "_" S] -> چ, [d "_" Z -> ج,
  [t "`"] -> ٹ, [d "`"] -> ڈ, [t "_" d] -> ت, [d "_" d -> د,
  p -> پ, b -> ب, [r "`"] -> ڑ, s -> س, [t "_" d] -> ت, r ->
  ر, h -> ہ, [t "_" d] -> ط, [t "_" d] -> ة, [s] -> ش, [s] ->
  ص];
read regex [[t "_" d 1] -> ط, [t "_" d 2] -> ة, [s 1] -> ش,
  [s 2] -> ص];
read regex [["_" h] -> ھ || [k | g | [t "_" S] | [d "_" Z] |
  [t "`"] | [d "`"] | [t "_" d] | [d "_" d] | p | b | [r
  "`"] | r | l | m | n | v] _ ];
read regex [I -> ِ, [A 1] -> ا, [A 1] -> ہ];
read regex [[A 1] -> ہ || [CONSONANTS] _ [.#.]];
compose net
```

Table 19: Sample XFST code for UIT to Punjabi/Shahmukhi scriptural translation

The code of Table 18 is deterministic and gives a unique output for a given Hindi text as we have shown in the example of Figure 19. On the other hand, the code of Table 19 is non-deterministic and parts of the code that cause the non-determinism are highlighted. This non-deterministic nature is also shown in the example of Figure 19. Commands 3 of Table 18, 6 and 8 of Table 19 are contextual mappings that we describe below.

Figure 20 shows graphically the final transducer for of the XFST code of Table 18.

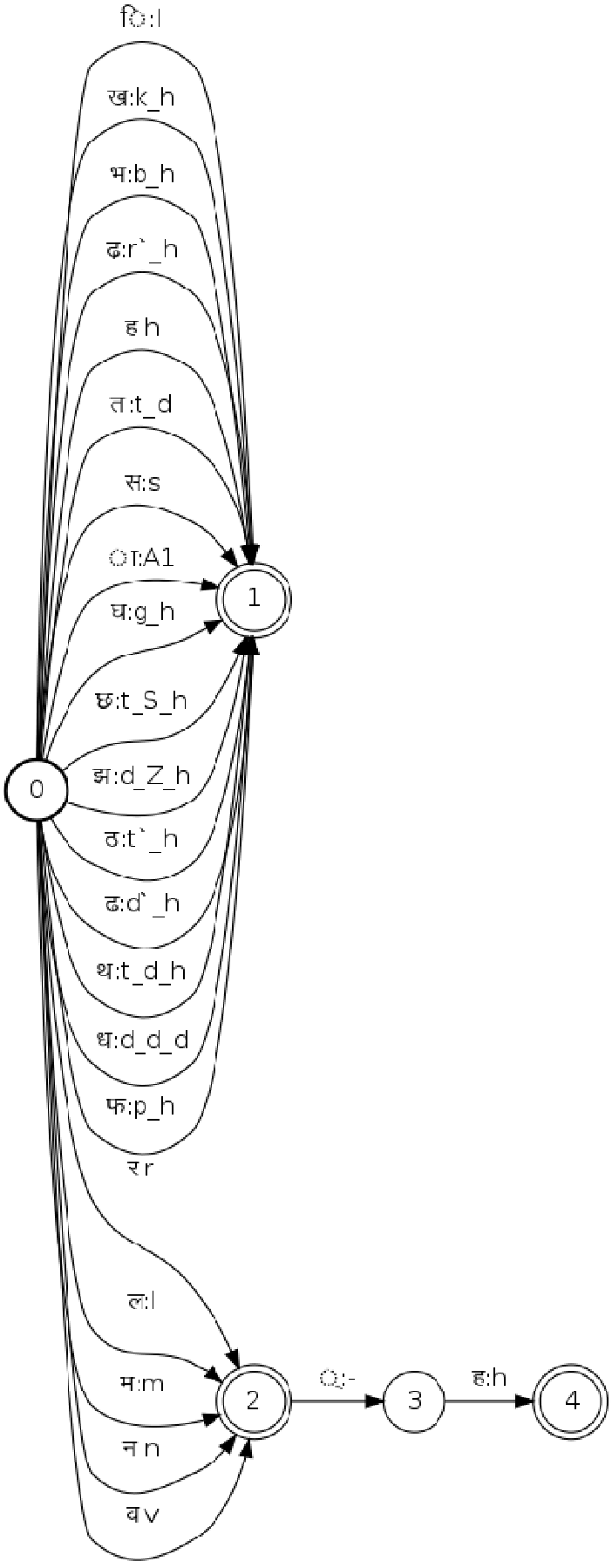


Figure 20. FST for XFST code of Table 18

2.3.1.1.2. Contextual mappings

A contextual mapping is a contextual rule that determines a desired output when a character appears in a certain context. XFST provides a contextual replace operator [89] for defining contextual replacement, shown in Figure 21.

```
Read regex [a -> b || c _ d ];
```

Figure 21: Contextual replace operator of XFST

The code shown in Figure 21 will be transformed by the XFST compiler into a contextual transducer that converts every occurrence of 'a' in the source by 'b' in the output when 'a' is preceded by 'c' and followed by 'd' in the source text. In XFST, it is possible to express context in the source and the target sides [12].

The third command of Table 18 models another contextual mapping saying that '़ह' is translated by '_h' when it is preceded by any of the characters र, ल, म, and न. The second last rule of Table 19 models the contextual mapping rule that 'A1' is translated into 'ہ' when it is at the end of a word and preceded by a consonant.

Vowel representations in Urdu, Punjabi/Shahmukhi, Sindhi, Seraiki/Shahmukhi and Kashmiri are highly context-sensitive, as shown in Table 11. These context-sensitive vowel representations can be expressed by context-sensitive transducers in XFST. A few examples of Urdu context-sensitive vowel conversions in UIT are given in Table 20 when vowels occur at the beginning of a word[26].

```
read regex [ا -> "@" || _ .#.];
read regex [[ِا] -> I, [ُا] -> U || _ .#.];
read regex [[ای] -> i, [اِی] -> i, [اَی] -> "{", [او] -> o,
  [اَو] -> O, [اُو] -> u, [اے] -> [e 3], [اَے] -> ["{" 3]
  || _ .#.];
```

Table 20: Sample XFST code for Urdu to UIT scriptural translation

XFST codes for the Indo-Pak languages are given in Annex 6.

2.3.1.2. Deterministic transducers

There are scriptural translational ambiguities as shown in Table 13 and accordingly our finite-state transducers are also ambiguous and non-deterministic. We can make these ambiguous transducers deterministic by removing certain choices and keeping only the most probable one. For example, UIT 's' can be converted into ث, س and ص in Urdu and Punjabi/Shahmukhi. We have conducted an analysis of a small corpus of 412,249 words and found interesting statistics for the characters of Table 13. They are given in Table 21. On the basis of these numbers, we can say that the most probable choice for 's' is س.

Hindi	Urdu (corpus word frequency)
त	ت (41,751), ط (1312)
स	س (53,289), ص (751), ث (86)
ह	ہ (72,850), ح (1800)
ज़	ز (2551), ض (1489), ذ (228), ظ (215), ژ (2)

Table 21: Corpus word frequncies of confusing characters [121]

Similarly, we can define default choices for other ambiguities and convert our ambiguous finite-state transducers into deterministic finite-state transducers. They are given in Annex 6.

[26] .#. marks the beginning or end of the input word or text when used as context before or after in the XFST code.

2.3.2. Finite-state model architecture

The system architecture of the finite-state model consists of Text Tokenizer, UIT Encoder, UIT Decoder, Text Generator and Finite-state Transducers. It is shown in Figure 22.

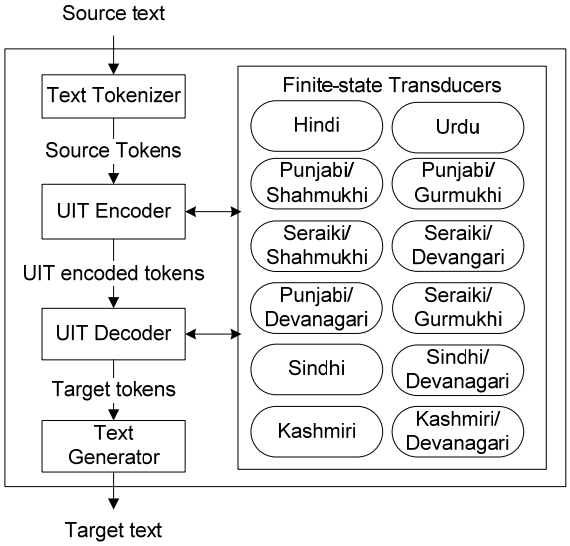


Figure 22: System architecture of the finite-state model for Indo-Pak scriptural translation

Text Tokenizer receives and converts the input source language text into constituent words or tokens. This list of the source language tokens is then passed to the UIT Encoder that encodes these tokens into a list of UIT tokens using the source language to UIT conversion transducer from the repertoire of Finite-State Transducers. These UIT tokens are given to the UIT Decoder that decodes them into target language tokens using the UIT to target language conversion transducer from the repertoire of Finite-State Transducers. Finally, the Text Generator generates the target language text from the translated target language tokens.

A sample run of this architecture on Hindi to Urdu scriptural translation on the example sentence of Figure 6 is shown in Table 22 (the Hindi input sentence is reproduced here in the first row). The UIT Decoder gives either a unique list of the target language tokens when deterministic finite-state transducers are used, or a list of sets of possible target language tokens for each input token when non-deterministic transducers are employed for decoding.

दुनिया को अमन की ज़रूरत है [dʊnɪjɑ ko əmən ki zərurət̪ hæ] (The world needs peace)			
Text Tokenizer	**UIT Encoder**	**UIT Decoder**	
		Unique output	*Ambiguous outputs*
दुनिया	dUnIjA1	دُنیا	[دُنیہ , دُنیا]
को	ko	کو	[قو , کو]
अमन	@mn	امن	[امن]
की	ki	کی	[قی , کی]
ज़रूरत	zrurt_d	زرُورت	[ژرُورت , ظرُورت , ذرُورت , ضرُورت , زرُرت , ژرُورط , ظرُورط , ذرُورط , ضرُورط , زرُرط , ژرُورة , ظرُورة , ذرُورة , ضرُورة , زرُرة]
है	h{	ہے	[ہے , ۓ]

Table 22: Sample run of finite-state model for scriptural translation from Hindi to Urdu

Text Generator converts the unique output of the UIT Decoder into an Urdu sentence with one error in the fifth word (highlighted), shown in Figure 23.

Figure 23: Unique output of the sample run by determintisc FSTs

On the other hand, from the ambiguous output of the UIT Decoder, we can generate 240 output sentences, but only one is the correct scriptural translation of the source Hindi sentence in Urdu. The correct sentence is shown in Figure 24. The sole difference between the output of the deterministic FST and the correct scriptural translation is highlighted in both sentences shown in Figure 23 and Figure 24.

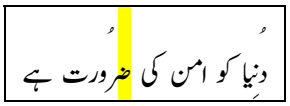


Figure 24: Correct scriptural translation of the sample run

Figure 25 shows a part of the confusion network for the ambiguous outputs of the non-deterministic finite-state model for scriptural translation.

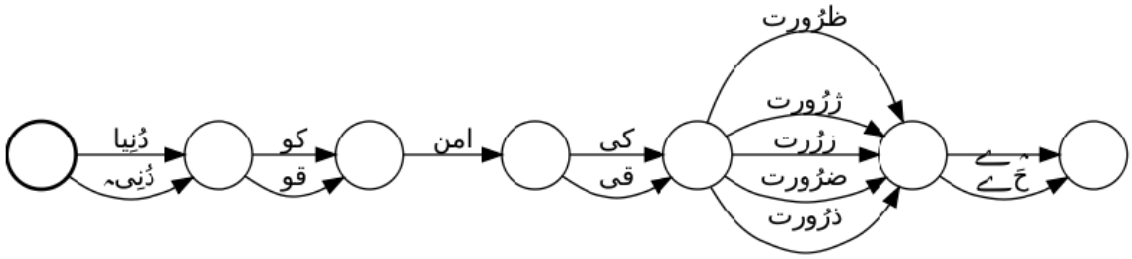


Figure 25: A part of confusion network for the ambiguous outputs of the non-deterministic FSTs

2.4. Experiments and results

The scriptural translation systems for Hindi–Urdu, Punjabi and Seraiki are freely available on Internet[27] with a simple *post-editing* facility like Google translate. By the post-editing, they not only correct the output for their use but also give us suggestions for the future improvements in the system.

For testing our finite-state systems, we have developed different test sets for different language/script pairs of the Indo-Pak languages. Our results are presented in the next two sections.

2.4.1. Test data

The Indo-Pak languages are under-resourced, but we were fortunate enough to find some raw monolingual corpora for Hindi, Urdu and Punjabi (Shahmukhi and Gurmukhi). We will describe the sources and details of these corpora in the next chapter. Here we will focus on the test sets that we have developed.

For the Hindi-Urdu pair, we have developed three test sets.

- The first Hindi-Urdu test set contains 52,753 Hindi–Urdu parallel words that we have extracted from Platts' Urdu, Hindi and English dictionary[28] [151].
- The second test set contains 200 sentences (4,281 words) of Hindi origin that are extracted at random from a Hindi corpus[29] of more than 3 million words.

[27] http://www.puran.info

[28] It was made available by Digital South Asian Library, University of Chicago.

[29] It is freely available http://www.cfilt.iitb.ac.in/ from IIT Bombay, India.

- The third test set contains 226 sentences (4,632 words) of Urdu origin that are extracted at random from the Urdu corpus[30] of more than 2 million words.

We will call these test sets "HU Test Set 1", "HU Test Set 2" and "HU Test Set 3" respectively.

It is a common practice in the Hindi community to use the characters क [k], ख [kʰ], ग [g], ज [ʤ], ड [ɖ], ढ [ɖʰ] and फ [p] instead of the characters क़ [q], ख़ [x], ग़ [ɣ], ज़ [z], ड़ [ɽ], ढ़ [ɽʰ] and फ़ [f] respectively, due to their shape similarities. In the HU Test Set 2, the extracted Hindi sentences were edited and corrected for these typo errors. Then, we translated the extracted Hindi sentences into Urdu by using our finite-state system for Hindi–Urdu scriptural translation. These translated Urdu sentences were post-edited for any error and completed by the necessary diacritical marks.

We have shown above that the diacritical marks are vital for Urdu to Hindi scriptural translation, but they are sparingly used by people in writing. To compute the performance of our finite-state model in this unfortunate but real situation, we developed another Urdu test data by removing diacritical marks from the post-edited 200 Urdu sentences. All these files will serve as input to our finite-state systems as well as an output reference for the automatic evaluations.

To build the HU Test Set 3, we first edited the extracted Urdu sentences for any error and restored the missing but necessary diacritics. We also developed a new Urdu test data without diacritics from the edited Urdu sentences by removing all diacritical marks from it. Then the edited Urdu sentences with diacritics were translated into Hindi using our finite-state system. The translated Hindi sentences were then post-edited for any error. Again all these data will serve both as an input as well as a reference output. Details on HU Test Set 1 are given in the next chapter.

For Punjabi/Shahmukhi and Punjabi/Gurmukhi, we started from the classical poetic work "ہیر وارث شاہ" by Waris Shah[31] [45] that we typed in 2004. We selected 500 couplets (5,069 words). This classical poetic work contained in total 14,223 words. During the preparation of this test data, we made sure that the Punjabi/Shahmukhi text contained the necessary diacritical marks. We developed a second Shahmukhi test set from it by removing the diacritical marks. Again, as for Hindi–Urdu, we converted the Shahmukhi text with diacritics into Gurmukhi using our finite-state system and the output was then post-edited for any error. All these files were used as input texts and as output references for both directions.

For Seraiki/Shahmukhi and Seraiki/Devanagari, we received a Seraiki book that was originally written in the derived Persio-Arabic script and was converted into Devanagari by using our online Hindi Urdu transliteration service[32]. The Seraiki/Shahmukhi character set is a superset of the Urdu script. So, Mr. Fareed Pirzada[33] first translated the book into Devanagari and then post-edited it to correct the missing characters and other errors. He shared the original and translated book with us. We also selected a test set of 500 sentences from this data. The Seraiki/Shahmukhi text does not contain any diacritical mark. We edited our test set and restored the necessary diacritical marks in it. As for other test sets, we developed another test data by removing the diacritical marks. It took us more than one month to develop these test sets.

Data set	Language pair	No. of words	No. of sentences	Source
HU Test Set 1	Hindi–Urdu	52,753	-	Platts dictionary
HU Test Set 2	Hindi–Urdu	4,281	200	Hindi corpus
HU Test Set 3	Hindi–Urdu	4,632	226	Urdu corpus
PU Test Set	Punjabi/Shahmukhi–Punjabi/Gurmukhi	5,069	500	Classical poetry
Seraiki Test Set	Seraiki/Shahmukhi–Seraiki/Devanagari	2,087	509	Seraiki poetry

Table 23: Data sets for testing scriptural translaiton

[30] A part is developed by the author and a part is obtained from EMILI

[31] A classical Punjabi poet and writer of 18th century

[32] http://www.puran.info/HUMT/index.html

[33] A native speaker of Seraiki from Lahore, Pakistan.

2.4.2. Results and discussion

For Urdu, Punjabi/Shahmukhi and Seraiki/Shahmukhi, we have two types of input data. One contains the necessary diacritical marks and the other does not contain any diacritical mark. On the other hand, Hindi, Punjabi/Gurmukhi and Seraiki/Devanagari have only one input data. Here we will report the results of our deterministic finite-state models.

For Hindi to Urdu scriptural translation, we have applied the finite-state model for Hindi to Urdu scriptural translation on all Hindi inputs of HU Test sets 1, 2 and 3. In general, it gives us an Urdu output with the necessary diacritical marks. To evaluate the performance of Hindi to Urdu scriptural translation of our finite-state system against the Urdu without diacritics, we have created a second Urdu output by removing all diacritical marks from the default Urdu output of the Hindi to Urdu finite-state model. We have calculated the *Word Accuracy* and *Sentence Accuracy* for the default and the processed Urdu outputs by comparing them with the Urdu references with and without diacritics respectively. To compute WAR and SAR, we have used the SCLITE utility from the Speech Recognition Scoring Toolkit (SCTK)[34] of NIST. The results of Hindi to Urdu scriptural translation are given in Table 24.

Test Set	Accuracy for default output		Accuracy for Processed output	
	Word Level	Sentence Level	Word Level	Sentence Level
HU Test Set 1	32.5%	-	78.9%	-
HU Test Set 2	90.8%	26.5%	91.0%	27%
HU Test Set 3	81.2%	8.8%	82.8%	9.7%

Table 24: Result of Hindi to Urdu scriptural translation by the finite-state system

Figure 26 shows a sample Hindi source text from HU Test set 2 of Hindi origin.

भारतीय सांस्कृतिक परंपराएँ कपिला वात्स्यायन कलाओं के क्षेत्र में सृजनात्मक कार्य प्रश्न बहन जी , आप का विद्याध्ययन प्रतिभासूचक रहा है और उस के बाद भारत सरकार के संस्कृतिविभाग के अध्यक्ष के नाते भी आप का कार्य उच्चकोटि का रहा है ;
इस पद पर कार्य करते हुए आप पर भारत के अन्तर्राष्ट्रीय सांस्कृतिक सम्बन्धों और भारत के भीतर सांस्कृतिक नीति के निर्माण का दायित्व रहा है , इस क्षेत्र में ऐसी भिन्नभिन्न चीज़ें आती हैं जैसे संग्रहालय , पुरातत्व , पुरालेख , ग्रंथालय और कलाएँ ;
केन्द्रीय स्तर पर भी और राज्य स्तर पर भी ।
आप का सरकारी कार्य तो भारत और विदेशों में सुविदित रहा है पर भारत की प्रदर्शनकारी और सुघट्य प्लास्टिक कलाओं के वर्गीकरण और व्याख्या करने के क्षेत्र में आप के सृजनात्मक योगदान के बारे में कम से कम औसत स्तर के शिक्षित सामान्य व्यक्ति को कुछ अधिक जानकारी नहीं है ।
क्या आप हमारे पाठकों के लिए भारतीय कलाओं और सौन्दर्यशास्त्र के क्षेत्र में किये गये अपने सृजनात्मक और अन्वेषणात्मक कार्य का संक्षिप्त विवरण देने की कृपा करें गी ?

Figure 26: A sample Hindi source text for Hindi to Urdu scriptural translation

The sample Hindi input text contains in total 169 words and 5 sentences.

Table 25 shows the default Urdu output text and the Urdu reference with diacritics for the sample Hindi input of Figure 26. Wrongly translated words are highlighted.

[34] http://www.itl.nist.gov/iad/mig//tools/

Urdu reference with diacritics	Default Urdu output
بھارتیّہ سانسکرتِک پرنپراءیں کپلا واتسیاین کلاءوں کے کشیتر میں سرجناتمک کاریہ پرشن بہن جی ، آپ کا وِدیادھیین پرتِبھاسُوچک رہاہَے اَور اُس کے بعد بھارت سرکار کے سنسکرتِوبھاگ کے ادھیکش کے ناتے بھی آپ کا کاریہ اُچکوٹی کا رہاہَے ؛ اِس پد پر کاریہ کرتے ہُوءے آپ پر بھارت کے اَتراشٹریّہ سانسکرتِک سمبندھوں اَور بھارت کے بھیتر سانسکرتِک بِیتی کے نِرمان کا دایِتو رہاہَے ، اِس کشیتر میں اَیسی بھِنّبھِنّ چیزیں آتی ہَیں جَیسے سنگرہالیہ ، پُرانتو ، پُرالیکھ ، گرنتھالیہ اَور کلاءیں ؛ کیندریّہ سْتر پر بھی اَور راجیہ سْتر پر بھی ۔ آپ کا سرکاری کاریہ تو بھارت اَور وِدیشوں میں سُودِت رہاہَے پر بھارت کی پردرشنکاری اَور سگھٹیہ پلاسٹِک کلاءوں کے ورگیکرن اَور ویاکھیا کرنے کے کشیتر میں آپ کے سرجناتمک یوگدان کے بارے میں کم سے کم اَوست سْتر کے شِکشِت سامانیہ ویکتی کو کچھ ادھِک جانکاری نہیں ہَے ۔ کیا آپ ہمارے پاٹھکوں کے لِءی بھارتیّہ کلاءوں اَور سَوندریشاستر کے کشیتر میں کیے گیے اپنے سرجناتمک اَور انویشناتمک کاریہ کا سنکشپت وِورن دینے کی کریا کریں گی ؟	بھارتیّہ سانسکرتِک پرنپراءیں کپلا واتسیاین کلاءوں کے کشیتر میں سرجناتمک کاریہ پرشن بہن جی ، آپ کا وِدیادھیین پرتِبھاسُوچک رہاہَے اَور اُس کے باد بھارت سرکار کے سنسکرتِوبھاگ کے ادھیکش کے ناتے بھی آپ کا کاریہ اُچکوٹی کا رہاہَے ؛ اِس پد پر کاریہ کرتے ہُوءی آپ پر بھارت کے اَتراشٹریّہ سانسکرتِک سمبندھوں اَور بھارت کے بھیتر سانسکرتِک بِیتی کے نِرمان کا دایِتو رہاہَے ، اِس کشیتر میں اَیسی بھِنّبھِنّ چیزیں آتی ہَیں جَیسے سنگرہالیہ ، پُرانتو ، پُرالیکھ ، گرنتھالیہ اَور کلاءیں ؛ کیندریّہ سْتر پر بھی اَور راجیہ سْتر پر بھی ۔ آپ کا سرکاری کاریہ تو بھارت اَور وِدیشوں میں سُودِت رہاہَے پر بھارت کی پردرشنکاری اَور سگھٹیہ پلاسٹِک کلاءوں کے ورگیکرن اَور ویاکھیا کرنے کے کشیتر میں آپ کے سرجناتمک یوگدان کے بارے میں کم سے کم اَوست سْتر کے شِکشِت سامانیہ ویکتی کو کچھ ادھِک جانکاری نہیں ہَے ۔ کیا آپ ہمارے پاٹھکوں کے لِءی بھارتیّہ کلاءوں اَور سَوندریشاستر کے کشیتر میں کیے گیے اپنے سرجناتمک اَور انویشناتمک کاریہ کا سنکشپت وِورن دینے کی کریا کریں گی ؟

Table 25: Default Urdu output of Hindi to Urdu scriptrual translation

Table 26 shows the processed Urdu output text and the Urdu reference without diacritics for the sample Hindi input of Figure 26.

Urdu reference without diacritics	Processed Urdu output
بھارتیہ سانسکرتک پرنپراءیں کپلا واتسیاین کلاءوں کے کشیتر میں سرجناتمک کاریہ پرشن بہن جی ، آپ کا ودیادھیین پرتبھاسوچک رہاہے اور اس کے بعد بھارت سرکار کے سنسکرتوبھاگ کے ادھیکش کے ناتے بھی آپ کا کاریہ اچکوٹی کا رہاہے ؛ اس پد پر کاریہ کرتے ہوءے آپ پر بھارت کے اتراشٹریہ سانسکرتک سمبندھوں اور بھارت کے بھیتر سانسکرتک بیتی کے نرمان کا دایتو رہاہے ، اس کشیتر میں ایسی بھنبھن چیزیں آتی ہیں جیسے سنگرہالیہ ، پرانتو ، پرالیکھ ، گرنتھالیہ اور کلاءیں ؛ کیندریہ سطر پر بھی اور راجیہ سطر پر بھی ۔ آپ کا سرکاری کاریہ تو بھارت اور ودیشوں میں سودت رہاہے پر بھارت کی پردرشنکاری اور سگھٹیہ پلاسٹک کلاءوں کے ورگیکرن اور ویاکھیا کرنے کے کشیتر میں آپ کے سرجناتمک یوگدان کے بارے میں کم سے کم اوست سطر کے شکشت سامانیہ ویکتی کو کچھ ادھک جانکاری نہیں ہے ۔ کیا آپ ہمارے پاٹھکوں کے لِءی بھارتیہ کلاءوں اور سوندریشاستر کے کشیتر میں کیے گیے اپنے سرجناتمک اور انویشناتمک کاریہ کا سنکشپت وورن دینے کی کریا کریں گی ؟	بھارتیہ سانسکرتک پرنپراءیں کپلا واتسیاین کلاءوں کے کشیتر میں سرجناتمک کاریہ پرشن بہن جی ، آپ کا ودیادھیین پرتبھاسوچک رہاہے اور اس کے باد بھارت سرکار کے سنسکرتوبھاگ کے ادھیکش کے ناتے بھی آپ کا کاریہ اچکوٹی کا رہاہے ؛ اس پد پر کاریہ کرتے ہوءی آپ پر بھارت کے اتراشٹریہ سانسکرتک سمبندھوں اور بھارت کے بھیتر سانسکرتک بیتی کے نرمان کا دایتو رہاہے ، اس کشیتر میں ایسی بھنبھن چیزیں آتی ہیں جیسے سنگرہالیہ ، پرانتو ، پرالیکھ ، گرنتھالیہ اور کلاءیں ؛ کیندریہ سطر پر بھی اور راجیہ سطر پر بھی ۔ آپ کا سرکاری کاریہ تو بھارت اور ودیشوں میں سودت رہاہے پر بھارت کی پردرشنکاری اور سگھٹیہ پلاسٹک کلاءوں کے ورگیکرن اور ویاکھیا کرنے کے کشیتر میں آپ کے سرجناتمک یوگدان کے بارے میں کم سے کم اوست سطر کے شکشت سامانیہ ویکتی کو کچھ ادھک جانکاری نہیں ہے ۔ کیا آپ ہمارے پاٹھکوں کے لِءی بھارتیہ کلاءوں اور سوندریشاستر کے کشیتر میں کیے گیے اپنے سرجناتمک اور انویشناتمک کاریہ کا سنکشپت وورن دینے کی کریا کریں گی ؟

Table 26: Processed Urdu output of Hindi to Urdu scriptrual translation

The finite-state scriptural translation system for Hindi to Urdu produces an Urdu output with diacritics. However, we know that the Urdu community is used to see the Urdu text without diacritics. Thus, we removed all diacritical marks from the Urdu output text that is more acceptable to the Urdu community. By this post-processing, we gain more than 40% accuracy in case of HU Test Set 1. We also gain in accuracy for the other test sets.

The sample processed Urdu output of Table 26 shows that post-processing has no effect on it. It also contains the same 9 error words as in Table 25, because none of these errors is caused by the absence of diacritical marks.

Our finite-state system produces its worst results on the HU Test Set 1. We have developed our translational rules by keeping in mind the necessary diacritical marks, while this test data is fully diacritized and contains a large number of diacritical marks. For this reason, our finite-state system gives a very poor result on this data. We can tackle this problem by incorporating rules of translation for a fully diacritized text.

The HU Test Set 2 gives better results than the HU Test Set 3 for Hindi to Urdu scriptural translation, because it is of Hindi origin and contains less Arabic and Persian words that cause problems in Hindi to Urdu scriptural translation. Also it contains fewer words that can cause transliterational or transcriptional ambiguities, as described previously. On average, the Hindi to Urdu scriptural translation is 80% and 20% accurate at word and sentence level respectively, using our finite-state system.

For the classification of our scriptural translation systems, we have devised two scales. One corresponds to the word accuracy and the other corresponds to the sentence level accuracy. They are shown in Figure 27 and Figure 28.

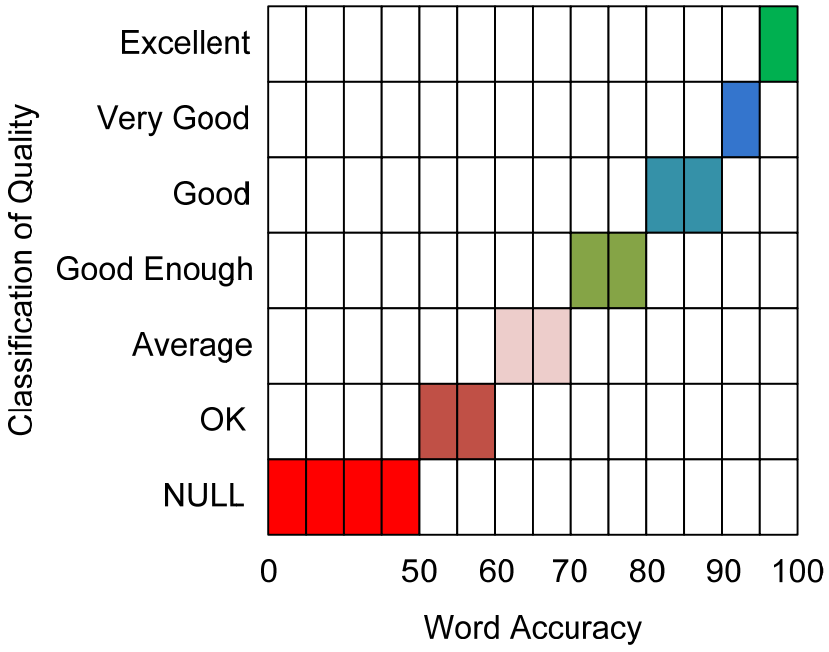


Figure 27: Classification scale based on the word accuracy for scriptural transaltion

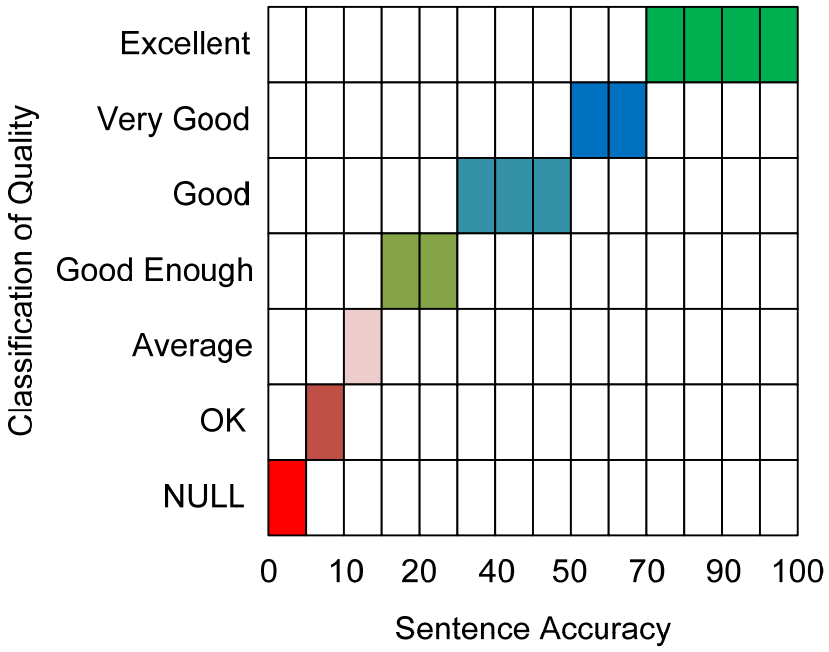


Figure 28: Classification scale based on the sentence accucary rate for scriptural translation

According to the scale of Figure 27 and Figure 28, the Hindi to Urdu scriptural translation system is classified in the 'Good' and the 'Good Enough' classes respectively.

The subjective evaluations like usability, effectiveness and adequacy depend on several factors. For example, the sample output Urdu texts of Table 25 and Table 26 show that only 9 Hindi words (out of 169 words) are wrongly translated into Urdu. On average, there are 1.8 errors per sentence and the translated Urdu text conveys the meaning very well. A user with a good knowledge of Hindi and Urdu languages would rate our Hindi to Urdu system quite high and would also rate the Urdu output very usable. Another user who wants to read a Hindi text, but does not

know Hindi, would also rate this system and the Urdu output quite high and very usable respectively, because it serves its purpose.

On the other hand, a user who wants to publish a Hindi book in Urdu, would rate this system not very good. This is because he has to localize the Hindi vocabulary of Sanskrit origin as the acceptance of the Hindi vocabulary in the Urdu community, target of his published book, is very low. Thus the subjective evaluation depends on various factors and it is not easy to compute such measures for the evaluation of a scriptural translation system.

For Urdu to Hindi scriptural translation, we have two inputs for each HU Test Set, but we have a single Hindi reference with which we will compare both Hindi outputs. We already know that it will give us less word accuracy for the Urdu input without diacritical marks that are mandatory for correct Urdu to Hindi scriptural translation. The results for Urdu to Hindi scriptural translation are given in Table 27.

Test Set	Accuracy for Urdu with Diacritics		Accuracy for Urdu without diacritics	
	Word Level	Sentence Level	Word Level	Sentence Level
HU Test Set 1	68.0%	-	31.2%	-
HU Test Set 2	83.9%	10%	53.0%	1%
HU Test Set 3	98.4%	73.9%	58.9%	0.4%

Table 27: Result of Urdu to Hindi scriptural translation by the finite-state system

For examples of Urdu to Hindi scriptural translation, the Urdu input texts with and without diacritical marks are shown in Table 25 and Table 26 (left columns) respectively as the Urdu reference texts. Table 28 shows the Hindi reference text with the Hindi scriptural translation of the Urdu input text with diacritical marks.

Hindi reference	Hindi output from the Urdu text with diacritics
भारतीय सांस्कृतिक परंपराएँ कपिला वात्स्यायन कलाओं के क्षेत्र में सृजनात्मक कार्य प्रश्न बहन जी , आप का विद्याध्ययन प्रतिभासूचक रहा है और उस के बाद भारत सरकार के संस्कृतिविभाग के अध्यक्ष के नाते भी आप का कार्य उच्चकोटि का रहा है ;	भारतीया सांस्करतिक परंपराएं कपिला वात्स्यायन कलाओं के क्शेत्र में सरजनात्मक कार्या प्रश्न बहन जी , आप का विद्याध्येन प्रतिभासूचक रहा है और उस के बाद भारत सरकार के संस्करतिविभाग के अध्यक्श के नाते भी आप का कार्या उच्चकोटी का रहा है;
इस पद पर कार्य करते हुए आप पर भारत के अन्तर्राष्ट्रीय सांस्कृतिक सम्बन्धों और भारत के भीतर सांस्कृतिक नीति के निर्माण का दायित्व रहा है , इस क्षेत्र में ऐसी भिन्नभिन्न चीज़ें आती हैं जैसे संग्रहालय , पुरातत्व , पुरालेख , ग्रंथालय और कलाएँ ;	इस पद पर कार्या करते हूए आप पर भारत के अन्तर्राश्ट्रीया सांस्करतिक सम्बन्धों और भारत के भीतर सांस्करतिक नीती के निर्मान का दायित्व रहा है , इस क्शेत्र में ऐसी भिन्नभिन्न चीज़ें आती हैं जैसे संग्रहालया , पुरातत्व , पुरालेख , ग्रंथालया और कलाएं;
केन्द्रीय स्तर पर भी और राज्य स्तर पर भी ।	केन्द्रीया स्तर पर भी और राज्या स्तर पर भी ।
आप का सरकारी कार्य तो भारत और विदेशों में सुविदित रहा है पर भारत की प्रदर्शनकारी और सुघट्य प्लास्टिक कलाओं के वर्गीकरण और व्याख्या करने के क्षेत्र में आप के सृजनात्मक योगदान के बारे में कम से कम औसत स्तर के शिक्षित सामान्य व्यक्ति को कुछ अधिक जानकारी नहीं है ।	आप का सरकारी कार्या तो भारत और विदेशों में सुविदित रहा है पर भारत की प्रदर्शकारी और सुघट्या प्लास्टिक कलाओं के वर्गीकरन और व्याख्या करने के क्शेत्र में आप के सरजनात्मक योगदान के बारे में कम से कम औसत स्तर के शिक्शित सामान्या व्यक्ती को कुछ अधिक जांकारी नहीं है ।
क्या आप हमारे पाठकों के लिए भारतीय कलाओं और सौन्दर्यशास्त्र के क्षेत्र में किये गये अपने सृजनात्मक और अन्वेषणात्मक कार्य का संक्षिप्त विवरण देने की कृपा करें गी ?	क्या आप हमारे पाठकों के लिये भारतीया कलाओं और सौन्दर्यशास्त्र के क्शेत्र में किये गए अपने सरजनात्मक और अन्वेशनात्मक कार्या का संक्शिप्त विवरन देने की करपा करें गी?

Table 28: Hindi output for Urdu input text with diacritics

For the Urdu input with diacritics, the accuracy of the Urdu to Hindi finite-state scriptural translation system is 83.9% at word level for HU Test Set 2 and it is classified as 'GOOD' the classification scale of Figure 27. On the other hand, it shows an accuracy of 10% for the same test set and is classified as 'AVERAGE' by the classification scale of Figure 28.

The average numbers of errors per sentence are 7.6 words, because the test set is of Hindi origin and it contains a large number of words of Sanskrit origin. The words of Sanskrit origin have a very bad effect on the finite-state scriptural translation system and the usability of the Hindi output would be rated less by the user in general. The classification of the scale of Figure 28 would correlate more accurately with the Hindi output usability and user satisfaction as compared to the classification of the scale of Figure 27.

Table 29 shows the Hindi output of the Urdu to Hindi scriptural translation for the Urdu input text without diacritics with its Hindi reference.

Hindi reference	**Hindi output from the Urdu text without diacritics**
भारतीय सांस्कृतिक परंपराएँ कपिला वात्स्यायन कलाओं के क्षेत्र में सृजनात्मक कार्य प्रश्न बहन जी , आप का विद्याध्ययन प्रतिभासूचक रहा है और उस के बाद भारत सरकार के संस्कृतिविभाग के अध्यक्ष के नाते भी आप का कार्य उच्चकोटि का रहा है ;	भारतया सांसकरतक परंपराएं कपला वातसयायन कलाओं के कशेतर में सरजनातमक कारया परशन बहन जी , आप का वदयाधेएन परतभासोचक रहा हे ओर अस के बाद भारत सरकार के संसकरतोभाग के अधेकश के नाते भी आप का कारया अचकोटी का रहा हे ;
इस पद पर कार्य करते हुए आप पर भारत के अन्तर्राष्ट्रीय सांस्कृतिक सम्बन्धों और भारत के भीतर सांस्कृतिक नीति के निर्माण का दायित्व रहा है , इस क्षेत्र में ऐसी भिन्नभिन्न चीज़ें आती हैं जैसे संग्रहालय , पुरातत्व , पुरालेख , ग्रंथालय और कलाएँ ;	अस पद पर कारया करते होए आप पर भारत के अंतराशटरया सांसकरतक समबंधों ओर भारत के भेतर सांसकरतक नेती के नरमान का दायतो रहा हे , अस कशेतर में एसी भंभन चेज़ें आती हें जेसे संगरहालया , पराततो , परालेख , गरंथालया ओर कलाएं ;
केन्द्रीय स्तर पर भी और राज्य स्तर पर भी ।	केंदरया सतर पर भी ओर राजया सतर पर भी ।
आप का सरकारी कार्य तो भारत और विदेशों में सुविदित रहा है पर भारत की प्रदर्शनकारी और सुघट्य प्लास्टिक कलाओं के वर्गीकरण और व्याख्या करने के क्षेत्र में आप के सृजनात्मक योगदान के बारे में कम से कम औसत स्तर के शिक्षित सामान्य व्यक्ति को कुछ अधिक जानकारी नहीं है ।	आप का सरकारी कारया तो भारत ओर वदेशों में सोदत रहा हे पर भारत की परदरशंकारी ओर सघटया पलासटक कलाओं के वरगेकरन ओर वयाखया करने के कशेतर में आप के सरजनातमक योगदान के बारे में कम से कम ओसत सतर के शकशत सामानया वेकती को कछ अधक जांकारी नहें हे ।
क्या आप हमारे पाठकों के लिए भारतीय कलाओं और सौन्दर्यशास्त्र के क्षेत्र में किये गये अपने सृजनात्मक और अन्वेषणात्मक कार्य का संक्षिप्त विवरण देने की कृपा करें गी ?	कया आप हमारे पाठकों के लये भारतया कलाओं ओर सोंदरेशासतर के कशेतर में कये गए अपने सरजनातमक ओर अनवेशनातमक कारया का संकशपत वोरन देने की करपा करें गी ?

Table 29: Hindi output for Urdu input text without diacritics

The Urdu to Hindi scriptural translation system is classified as 'OK' by the scale of Figure 27 for HU Test set 2 and 3. It is classifies as 'NULL' for HU Test Set 1. It is also classified as 'NULL' by the scale of Figure 28 for all three test sets. On average, there exist 17.6 errors per sentence. In this case, the usability of Hindi output is also very bad. The errors in Table 29 are due to the vocabulary of Sanskrit origin and the absence of diacritical marks in the Urdu input text. Therefore the vocabulary diversity and the missing information have a large affect on the accuracy of the Hindi–Urdu scriptural translation and the finite-state system is classified as 'NULL' in terms of usability, effectiveness and user satisfaction.

For Punjabi scriptural translation, we have applied our finite-state model on the Gurmukhi input. Like Hindi to Urdu translation, it also gives Punjabi/Shahmukhi output with necessary diacritics and we developed a processed output by removing all diacritics from it. We have computed WAR and SAR results by comparing these two outputs with the Punjabi/Shahmukhi reference texts with and without diacritics.

In the reverse direction, we have two types of input Punjabi/Shahmukhi texts, one with diacritics and the other without diacritics. The results of the Punjabi finite-state scriptural translation systems are given in Table 30.

	Accuracy for default output		**Accuracy for processed output**	
Punjabi/Gurmukhi to Punjabi/Shahmukhi	*Word level*	*Sentence level*	*Word level*	*Sentence level*
	84.2%	27.8%	85.2%	29.9%

	Accuracy for Shahmukhi with diacritics		**Accuracy for Shahmukhi without diacritics**	
Punjabi/Shahmukhi to Punjabi/Gurmukhi	*Word level*	*Sentence level*	*Word level*	*Sentence level*
	98.8%	90.3%	67.3%	6.4%

Table 30: Results of Punjabi scriptural translation by the finite-state system

Compared to the Hindi–Urdu pair, the Punjabi/Shahmukhi–Punjabi/Gurmukhi pair is computationally less hard. The post-processing to the default out of the finite-state scriptural translation systems for Punjabi/Gurmukhi to Punjabi/Shahmukhi also helps to gain an increase of approximately 1% and 2% at word and sentence levels respectively. The Punjabi/Shahmukhi to Punjabi/Gurmukhi scriptural translation system is classified as 'GOOD' by both scales of Figure 27 and 28. Thus the usability of the Punjabi finite-state scriptural translation system is higher than the Hindi–Urdu finite-state scriptural translation system.

In the reverse direction, the Punjabi/Shahmukhi to Punjabi/Gurmukhi scriptural translation system gives an accuracy of 98.8% and 67.3% for the Punjabi/Shahmukhi input text with and without diacritics respectively. For the Punjabi/Shahmukhi input text with diacritics, the scriptural translation system is classified as 'EXCELLENT' by both scales of Figure 27 and Figure 28. On the other hand, it is classified as 'NULL' according to the scale of Figure 28 for the Punjabi/Shahmukhi input text without diacritical marks.

Similar to Hindi–Urdu and Punjabi finite-state scriptural translation, we have applied our finite-state systems to the Seraiki test set. Here also, we have also developed a processed Seraiki/Shahmukhi output from the default output of our finite-state system by removing the diacritics. The results are given in Table 31.

	Accuracy for default output		**Accuracy for processed output**	
Seraiki/Devanagari to Seraiki/Shahmukhi	*Word level*	*Sentence level*	*Word level*	*Sentence level*
	81.3%	19.4%	83.7%	20.3%

	Accuracy for Shahmukhi with diacritics		**Accuracy for Shahmukhi without diacritics**	
Seraiki/Shahmukhi to Seraiki/Devanagari	*Word level*	*Sentence level*	*Word level*	*Sentence level*
	95.2%	76.4%	58.6%	8.6%

Table 31: Results of Seraiki scriptural translation by the finite-state system

In the case of the Seraiki/Devanagari to Seraiki/Shahmukhi scriptural translation system, the post-processing also helps to gain an increase in word accuracy of approximately 1 to 2 percent both at the word and the sentence levels. The accuracy for both the default and the processed Seraiki/Shahmukhi outputs is also more than 80% at word level. The system is classified as 'GOOD' and 'GOOD ENOUGH' according to the scale of Figure 27 and Figure 28 respectively.

The absence of diacritical marks in the Seraiki/Shahmukhi has a very bad effect on the accuracy of the finite-state scriptural translation system. The scriptural translation system is classified as 'NULL' for the Seraiki/Shahmukhi input text without diacritics.

The above results support our hypothesis that lack of important information and of diacritical marks in the source texts considerably lowers the best possible quality of the scriptural translation of the Indo-Pak languages. They are crucial and their absence in the input texts decreases the performance considerably, from more than 80% to less than 60% at word level. Also, the translation system classification changes from 'GOOD' to 'NULL'. It is an unfortunate but a real situation for Indo-Pak scriptural translation. Thus, restoring the missing information and the diacritical marks in the input texts, or minimizing their effect on the scriptural translation of the Indo-Pak languages is a great challenge and a complex problem.

We have also computed the BLEU and NIST scores for all these test sets. They are given in Table 32. Classifications according to the scales of Figure 27 are also shown in the table.

Hindi to Urdu Scriptural Translation				
Test Set	**BLEU**		**NIST**	
	Default Urdu Output	Processed Urdu Output	Default Urdu Output	Processed Urdu Output
HU Test Set 1	0.2935 (NULL)	0.7893 (GOOD ENOUGH)	3.2778	8.7144
HU Test Set 2	0.8375 (GOOD)	0.8402 (GOOD)	10.5823	10.6030
HU Test Set 3	0.5823 (OK)	0.6112 (AVERAGE)	8.9508	9.1513

Urdu to Hindi Scriptural Translation				
Test Set	**BLEU**		**NIST**	
	Input with Diacritics	Input without Diacritics	Input with Diacritics	Input without Diacritics
HU Test Set 1	0.6789 (AVERAGE)	0.3113 (NULL)	7.6229	3.4706
HU Test Set 2	0.6372 (AVERAGE)	0.1817 (NULL)	9.1702	4.8751
HU Test Set 3	0.9607 (EXCELLENT)	0.2435 (NULL)	11.8271	5.9887

Punjabi/Gurmukhi to Punjabi/Shahmukhi Scriptural Translation				
Test Set	**BLEU**		**NIST**	
	Default Output	Processed Output	Default Output	Processed Output
PU Test Set		0.6709 (AVERAGE)		8.8752

Punjabi/ Shahmukhi to Punjabi/Gurmukhi Scriptural Translation				
Test Set	**BLEU**		**NIST**	
	Input with Diacritics	Input without Diacritics	Input with Diacritics	Input without Diacritics
PU Test Set	0.9672 (EXCELLENT)	0.3592 (NULL)	10.5464	6.6469

Table 32: BLEU and NIST scores for scriptural translation of Hindi - Urdu and Punjabi

2.5. Conclusion

Finite-state methods are robust and efficient to implement scriptural translation rules in a very precise and compact manner. Especially XFST provides us a very simple and easy SLLP for developing finite-state transducers as compared to the other available finite-state platforms.

The missing information and the diacritical marks in all the source texts of the Indo-Pak languages proved to be very critical, crucial and important for achieving high and accurate results. Thus restoration of the missing information and the diacritical marks or reducing the effect of their absence on the scriptural translation of the Indo-Pak languages is one of the major questions for further study and work.

Part II
Statistical, Hybrid and Interactive Models for Scriptural Translation

Chapter 3. Empirical and Hybrid Methods for Scriptural Translation

Statistical Machine Translation (SMT) [32, 33, 102, 105, 115] is a big leap in the field of MT, but most of time SMT lingers with the data scarcity problem, especially in the case of under-resourced languages or language pairs. Empirical and statistical approaches are also getting attention of researchers for solving the problems of transliteration and/or transcription [1, 3, 4, 46, 50, 52, 65, 68, 71, 97, 112, 120, 124, 130, 137, 140, 143, 144, 146, 148, 164, 173]. Following them, we have also developed statistical models for solving the problem of scriptural translation for the Indo-Pak languages.

We found the data of a Urdu-Hindi-English dictionary [151] from Digital South Asia Library (DSAL), University of Chicago[35] for Hindi–Urdu pair (the only almost parallel data, we found). Using these parallel and monolingual resources, we can check the usability of empirical methods for scriptural translation. First we will describe the SMT model developed for Hindi–Urdu scriptural translation. Then we describe a hybrid model, a combination of FST and statistical methods, to utilize monolingual resources for the improvement of scriptural translation performance. Finally, we compare different models used for scriptural translation.

3.1. SMT Model for Scriptural Translation

SMT is becoming a fashion in the field of MT because of its robustness and efficiency. It is easy and very fast to construct an SMT system when large enough parallel data are available, using existing freeware tools like GIZA++ [141], Moses [105] and SRILM [175]. For developing an SMT system, we prepare the parallel data (if available) for alignment with GIZA++ that generates different types of alignment like those of Hidden Markov Model (HMM), IBM model 3, 4 and 5 alignments. We train or develop a *translation model(s)* from the aligned data using the Moses decoder. We also develop *target language model(s)*. Then by the serial combination of the translation model and the target language model(s), we develop an SMT system. Finally, we validate the translation system with the help of the test data and the target reference data.

For example, consider a source language, say French, f, and a target language, say English, e. The SMT system from French to English is modeled as a conditional probability tion $\mathrm{P}(e|f)$. By using Bayes' theorem, it becomes:

$$\mathrm{P}(e|f) = \frac{\mathrm{P}(f|e)\mathrm{P}(e)}{\mathrm{P}(f)} \qquad (1)$$

Where $\mathrm{P}(f|e)$ is the translation model and $\mathrm{P}(e)$ is the target language model. Finally, we arrive at the fundamental equation of *source channel approach* for SMT [32].

$$\hat{e} = \underset{e}{\operatorname{argmax}}\, \mathrm{P}(e)\mathrm{P}(f|e) \qquad (2)$$

[35] We are thankful to Mr. James Nye, project director of Digital South Asia Library (DSAL), University of Chicago for sharing data with us. The original data from DSAL does not contain the Urdu words in the Perso-Arabic script and instead contains a roman transcription for Urdu words. We automatically generated the Urdu words and returned the modified data to DSAL.

Here $\hat{e}$ has to maximize the product of $P(e)$ and $P(f|e)$ [33].

Following this SMT fashion, we have also developed SMT models for scriptural translation for Hindi–Urdu using the Urdu–Hindi–English dictionary data [151].

3.1.1. Training data

In general SMT, the training data consist of parallel sentences of the converted language pair. In our case, there exist no such data. We only have an Urdu-Hindi-English dictionary [151]. DSAL has digitized this dictionary and has developed a web interface[36] to access the dictionary. The original dictionary contains 55,253 entries. The original published dictionary contains the Persio-Arabic transcriptions of all word entries. On the other hand, it contains Hindi/Devanagari transcriptions only for words that are not of Persio-Arabic origin.

In the digital data of DSAL, dictionary entries did not contain the Urdu words in the Persio-Arabic script. Instead, they contained Roman transcriptions. As for the original dictionary, these data also do not contain Hindi/Devanagari transcriptions for words of Persio-Arabic origin.

While providing the digital data in March 2007, DSAL asked us to give them back the processed data with Urdu script. After finishing the initial processing in April 2007, we sent back the processed data with Urdu script to DSAL. It took us 15 days to finish the initial processing without any post-editing that was required for cleaning the data for some errors. DSAL has updated the online Platts' dictionary in August 2008 by adding the Urdu script that was not present before.

Sample entries are shown in Figure 29. We are interested in the highlighted parts that contain information about the Hindi and Urdu words of an entry.

<div2 type="article" id="ābādī"><head><hi>ābādī</hi></head> P
<pa>ābādī</pa> ābādī, s.f. Inhabited spot or place; colony; population, number of inhabitants; cultivated place;cultivation; the part of a village lands brought under cultivation; increased assessment (=beshī); prosperity; stateof comfort; happiness, joy, pleasure. </div2>
<div2 type="article" id="अबार_abār"><head><hi>अबार abār</hi></head> H
<pa>abār</pa> अबार abār [S. अवार], s.m. This side, the nearbank of a river. </div2>

Figure 29: Sample Entries from [151]

The format of the data is not strictly XML-based, but contains certain tags. For example, for the Hindi transcription of the headword, it uses the tag <hi></hi>. Similarly, it uses the tag <pa></pa> for the Persio-Arabic transcription. If we examine the second entry of our example in Figure 29, the tag <hi></hi> contains not only the Hindi transcription, but also the Roman transcription. Thus, during the process of extraction of the Hindi and the Roman transcriptions, we have to handle such cases carefully.

We have developed a program that parses the data and extracts Hindi and Roman transcriptions from each dictionary entry. After the extraction phase, we do an exhaustive analysis of the extracted Roman transcriptions to build finite-state transducers that can generate Urdu words from them and can also generate Hindi words, if missing. These transducers are given in Annex 6. It took us approximately 250 hours of work to develop our Hindi–Urdu parallel lexicon of 55,253 words. Here Hindi–Urdu parallel words means that the same word written in Hindi/Devanagari and Urdu/Urdu (derived Persio-Arabic script). We have used 50,000 Hindi-Urdu parallel words as training data to train our statistical scriptural translation models for Hindi-Urdu. First of all, we need to align the Hindi-Urdu parallel data.

[36] http://dsal.uchicago.edu/dictionaries/platts/

3.1.2. Data alignments

We can align the Hindi-Urdu parallel lexicon with different strategies.

- We can align the data at character-level by considering each Hindi-Urdu parallel word pair as a parallel sentence pair and each character in the 2 words as a word in a sentence of a parallel sentence pair (Hindi-Urdu words).
- During the analysis of character-level alignments, we found that we can improve the alignments by grouping some specific sequences of characters into clusters. We call such alignments *cluster alignments*.
- We can also align the data at syllable level. For this alignment, first we need to syllabify each Hindi-Urdu parallel word into constituent syllables. Then we can align the parallel data at syllable level by considering syllables in the parallel entry as parallel words.

Here we describe each alignment strategy one by one.

3.1.2.1.1. Character alignments

Table 33 shows some Hindi-Urdu parallel words with their phonetic representations and English glosses. The first two columns show the Hindi-Urdu parallel words.

Hindi	Urdu	IPA	English
अब्ब	اَبّا	əbbɑ	Father
इबलाग़	اِبلاغ	ɪbəlɑx	Conveying
उबलाना	اُبلانا	ʊbəlɑnɑ	To boil
इबलीस	اِبلیس	ɪbəlis	Devil
अभागेपन	ابھاگپن	əbʰɑgepən	Unfortunate
अप्रैल	اپریل	əpræl	April
अच्छा	اچھا	əʧʧɑ	Good

Table 33: Examples from a Hindi-Urdu parallel lexicon

The Urdu word list that we have developed from the Roman transcriptions of the Urdu-Hindi-English dictionary [151] using finite-state transducers contains all required diacritics.

We have previously shown that diacritical marks are the backbone of the vowel system in Urdu and that they are mandatory for the correct pronunciation of an Urdu word, as well as for scriptural translation and Urdu NLP. To model this unfortunate but real situation, we developed another Urdu word list that does not contain any diacritic by removing all diacritics from the fully diacritized Urdu word list. In this way, we have developed two sorts of Hindi-Urdu parallel data.

For *character alignment*, we put a space after each character in the Hindi and Urdu (whether diacritized or not) words and prepare the Hindi and Urdu parallel data for character alignment. Example words of Table 34 prepared for character alignment are shown in Table 34.

Hindi	Urdu with diacritics	Urdu without diacritics
अ ब ् ब	ا ب ْ ب ّ ا	ا ب ا
इ ब ल ा ग़	اِ ب ل ا غ	ا ب ل ا غ
उ ब ल ा न ा	اُ ب ل ا ن ا	ا ب ل ا ن ا
इ ब ल ी स	اِ ب ل ِ ی س	ا ب ل ی س
अ भ ा ग े प न	اَ ب ھ ا گ ی پ ن	ا ب ھ ا گ ی پ ن
अ प ् र ै ल	اَ پ ْ ر َ ی ل	ا پ ر ی ل
अ च ् छ ा	ا ٓ چ ھ ّ ا	ا چ ھ ا

Table 34: Examples of Hindi - Urdu parallel words

We have developed two types of alignment data for character alignment.

- Hindi and Urdu with diacritics alignment (Hi-UrWD)
- Hindi and Urdu without diacritics alignment (Hi-UrWOD)

GIZA++ gives us two types of alignments (1) *source to target* and (2) *target to source*. In our case, GIZA++ gives us (1) Hindi to Urdu with diacritics and (2) Urdu with diacritics to Hindi alignments for Hindi and Urdu with diacritics alignment data. Similarly, it gives us another two alignments for Hindi and Urdu without diacritics alignment data. Table 35 shows Hindi to Urdu with diacritics alignments of our example words of Table 34.

1	# Sentence pair (6) source length 4 target length 5 alignment score : 1.70006e-05 अ ब ु ब ा NULL ([ ]) 5]) ا ([3]) ُ([4 2]) ب ([1]) ا])
2	# Sentence pair (114) source length 6 target length 5 alignment score : 0.00032306 इ ब ल ा ग़ NULL ([ ]) 5]) غ ([4]) ا ([3]) ل ([2]) ب ([ ]) ِ([1]) ا])
3	# Sentence pair (115) source length 7 target length 6 alignment score : 0.000154595 उ ब ल ा न ा NULL ([ ]) ([5]) ن ([4]) ا ([3]) ل ([2]) ب ([ ]) ُ([1]) ا 6]) ا])
4	# Sentence pair (128) source length 7 target length 5 alignment score : 5.58545e-05 इ ब ल ी स NULL ([ ]) س ([4]) ی ([ ]) ِ([3]) ل ([2]) ب ([ ]) ِ([1]) ا 5])])
5	# Sentence pair (167) source length 9 target length 7 alignment score : 3.20243e-05 अ भ ा ग े प न NULL ([ ]) 5]) ی ([4]) گ ([3]) ا ([ ]) ھ ([2]) ب ([1]) ا 7]) ن ([ ]) ُ([6]) پ ([ ])
6	# Sentence pair (464) source length 6 target length 6 alignment score : 7.74271e-06 अ प ु र ै ल NULL ([ ]) 6]) ل ([5]) ی ([3]) ُ([4]) ر ([2]) پ ([1]) ا])
7	# Sentence pair (1183) source length 5 target length 5 alignment score : 3.13657e-05 अ च ु छ ा NULL ([ ]) 5]) ا ([3]) ُ([4]) ھ ([2]) چ ([1]) ا])

Table 35: Character alignment examples from Hindi to Urdu with diacritics

3.1.2.1.2. Cluster alignments

Alignment plays a critical role in SMT [56, 58, 70, 74, 107, 116]. It is a key step for building an SMT system. The quality of parallel data and the word alignment have a significant impact on learning the translation model and consequently on the quality of the SMT system [56, 58, 74]. It is always better do an analysis of the alignment and correct the alignment errors to reduce the Alignment Error Rate (AER).

We also analyzed the alignments produced by GIZA++. We found that we can improve our alignments to reduce the AER. The incorrect alignments are highlighted in Table 35. We have already described the complex and highly contextual nature of vowel systems in Urdu, Punjabi/Shahmukhi, Sindhi, Seraiki/Shahmukhi and Kashmiri/Urdu (see pages 26-27). In the second row of Table 35, the Hindi vowel इ [ɪ] is aligned with ALEF (ا) and ZER (ِ) is aligned to NULL. The alignment is not completely incorrect, but the vowel इ [ɪ] must be aligned with both

ALEF (ا) and ZER (ِ). Similarly, the Hindi vowel उ [ʊ] must be aligned with ALEF (ا) and PESH (ُ) in the third row. In these examples, one character of Hindi must be aligned with a sequence of characters in Urdu. Interestingly, we have observed that GIZA++ correctly aligns such cases for Urdu (with or without diacritics) to Hindi alignment. The correct Urdu to Hindi GIZA++ alignments of these examples are shown in Table 36.

```
# Sentence pair (114) source length 5 target length 6 alignment score
: 0.00150584
ا ِ ب ل ا غ
NULL ([ ]) इ ([ 1 2 ]) ब ([ 3 ]) ल ([ 4 ]) ा ([ 5 ]) ग़ ([ 6 ])
# Sentence pair (115) source length 6 target length 7 alignment score
: 0.00135167
ا ُ ب ل ا ن ا
NULL ([ ]) उ ([ 1 2 ]) ब ([ 3 ]) ल ([ 4 ]) ा ([ 5 ]) न ([ 6 ]) ा ([ 7
])
```

Table 36: Vowel alignment from Urdu with diacritics to Hindi

All such vowel alignments can be improved by clustering the specific sequences of characters in Urdu side. These Urdu character sequences are listed in Table 37 with their equivalent Hindi characters.

Urdu	Hindi	Urdu	Hindi	Urdu	Hindi
ا + ◌َ = اَ	अ [ə]	ع + ◌َ = عَ	अ [ə]	ا + ◌َ + و = اَو	औ [ɔ]
ا + ◌ُ = اُ	उ [ʊ]	ع + ◌ُ = عُ	उ [ʊ]	◌ِ + ی = ◌ِی	ी [i]
ا + ◌ِ = اِ	इ [ɪ]	ع + ◌ِ = عِ	इ [ɪ]	◌ُ + و = ◌ُو	ू [u]
ا + ی = ای	ए [e]	ع + ی = عی	ए [e]	ء + ی = ءی	ई [i]
ا + ے = اے	ए [e]	ع + ے = عے	ए [ə]	ء + ے = ءے	ए [e]
ا + ◌َ + ی = ایَ	ऐ [æ]	ع + ◌َ + ی = عیَ	ऐ [æ]	ع + ◌َ + و = عوَ	औ [ɔ]
ا + ◌َ + ے = اےَ	ऐ [æ]	ع + ◌َ + ے = عےَ	ऐ [æ]	◌َ + ی = ◌َی	ै [æ]
ا + ◌ِ + ی = اِی	ई [i]	ع + ◌ِ + ی = عِی	ई [i]	◌َ + و = ◌َو	ॉ [ɔ]
ا + و = او	ओ [o]	ع + و = عو	ओ [o]	ء + و = ءو	ओ [o]
ا + ◌ُ + و = اُو	ऊ [u]	ع + ◌ُ + و = عُو	ऊ [u]	◌َ + ے = ◌َے	ै [æ]

Table 37. Urdu character sequeces for vowel alignments

In the case of consonants, we also observed few alignment problems. In Urdu, *gemination*[37] of a consonant is marked with SHADDA (ّ), while in Hindi, it is written as a *conjunct* form. The highlighted alignments of the first row of Table 35 align the Hindi characters ब [b] and ् with the Urdu characters BEH (ب) and SHADDA (ّ) respectively. Although in the Hindi to Urdu alignment, this alignment is not completely wrong, but this geminated consonant alignment problem is more evident for Urdu to Hindi alignment where a Hindi consonant is aligned with NULL. An example of Urdu to Hindi alignment is shown in Table 38.

```
# Sentence pair (754) source length 9 target length 10 alignment score
: 8.0263e-13
ا ِ تْ ِ ف ا ق اً
NULL ([ 9 ]) इ ([ 1 2 ]) त ([ 3 ]) ् ([ 4 ]) त ([ ]) ि ([ 5 ]) फ़ ([ 6
]) ा ([ 7 ]) क़ ([ 8 ]) न ([ 10 ])
```

Table 38. Gemination alignment from Urdu to Hindi

[37] Gemination happens when a spoken consonant is pronounced for an audibly longer period of time than a short consonant. (http://en.wikipedia.org/wiki/Gemination)

If the Hindi character sequence ब + ् + ब = ब्ब [bb] is aligned with the Urdu character sequence ب + ّ = بّ [bb], then it is a completely correct and valid alignment.

We also observed alignment problems for aspirated consonants because they are represented by a sequence of characters in Urdu and by either a single character or a sequence of characters in Hindi. For Hindi to Urdu alignment, this problem is highlighted in row 5 and 7 of Table 35. For Urdu to Hindi alignment, an example is shown in Table 39.

```
# Sentence pair (1183) source length 5 target length 5 alignment score
: 8.57561e-05
ا چ ھ َ ا
NULL ([ ]) अ ([ 1 ]) च ([ 2 ]) ् ([ ]) छ ([ 3 4 ]) ा ([ 5 ])
```

Table 39. Aspirated consonant alignment from Urdu to Hindi

All these problems increase the AER. Thus we decide to cluster such sequences for improving the alignment and therefore the quality of scriptural translation between Hindi and Urdu. We have developed finite-state transducers for performing the clustering operation on our Hindi Urdu parallel data. These transducers are given in Annex 6. Examples of words obtained after the clustering operation are given in Table 40 with their IPA representations. The third column's IPA representations are for both Hindi and Urdu with diacritics entries.

Hindi	**Urdu with diacritics**	**IPA**	**Urdu without diacritics**	**IPA**
अ ब्ब	ا بّ ا	ə bb ɑ	ا ب ا	ə b ɑ
इ ब ल ा ग़	اِ ب ل ا غ	ɪ b l ɑ x	ا ب ل ا غ	ə b l ɑ x
उ ब ल ा न ा	اُ ب ل ا ن ا	ʊ b l ɑ n ɑ	ا ب ل ا ن ا	ə b l ɑ n ɑ
इ ब ल ी स	اِ ب ل ِی س	ɪ b l i s	ا ب ل ی س	ə b l e s
अ भ ा ग े प न	اَ بھ ا گ ی پ ن	ə bʰ ɑ g e pə n	ا بھ ا گ ی پ ن	ə bʰ ɑ g e pə n
अ प ् र ै ल	اَ پ ْ ر َی ل	ə p r æ l	ا پ ر ی ل	ə p r e l
अ च्छ ा	اَ چھّ ا	ə ʧʧʰ ɑ	ا چھ ا	ə ʧʰ ɑ

Table 40: Hindi - Urdu example words for alignment

After running GIZA++ on the cluster Hindi-Urdu parallel data, we corrected the alignment errors that are discussed above. The result is shown in Table 41.

1	# Sentence pair (6) source length 3 target length 3 alignment score : 0.0214204 अ ब्ब ा NULL ([ ]) 3]) ا ([2]) بّ ([1]) ا])
2	# Sentence pair (114) source length 5 target length 5 alignment score : 0.0942275 इ ब ल ा ग़ NULL ([ ]) 5]) غ ([4]) ا ([3]) ل ([2]) ب ([1]) إ])
3	# Sentence pair (115) source length 6 target length 6 alignment score : 0.0373352 उ ब ल ा न ा NULL ([ ]) 6]) ا ([5]) ن ([4]) ا ([3]) ل ([2]) ب ([1]) أ])
4	# Sentence pair (128) source length 5 target length 5 alignment score : 0.0430949 इ ब ल ी स NULL ([ ]) 5]) س ([4]) ِی ([3]) ل ([2]) ب ([1]) إ])
5	# Sentence pair (167) source length 8 target length 7 alignment score : 0.000313045 अ भ ा ग े प न NULL ([ ]) 6]) پ ([5]) ی ([4]) گ ([3]) ا ([2]) ہِ ([1]) ا 7]) ن ([ ]) ً ([ ])
6	# Sentence pair (464) source length 5 target length 6 alignment score : 1.71945e-05 अ प ् र ै ल NULL ([3]) 6]) ل ([5]) ًی ([4]) ر ([2]) پ ([1]) ا])
7	# Sentence pair (754) source length 8 target length 7 alignment score : 0.000371183 इ त ि फ़ ा क़ न NULL ([ ]) 6]) ق ([5]) ا ([4]) ف ([3]) ِ([2]) تَ ([1]) إ 7]) ً([ ]) ا ([ ])
8	# Sentence pair (1183) source length 3 target length 3 alignment score : 0.0207299 अ च्छ ा NULL ([ ]) 3]) ا ([2]) ھَچ ([1]) ا])

Table 41. Cluster alignments examples from Hindi to Urdu with diacritics

It seems that better cluster alignments will help to learn a good quality translation model and accordingly will enhance the accuracy of our SMT systems.

3.1.3. Translation models

Based on two types of *character* and *cluster* alignments of our Hindi-Urdu (with or without diacritics) parallel word lists, we have developed eight different translation models $P(e|f)$ using the Moses toolkit [105].

- Translation model learned from Hindi to Urdu with diacritics character alignment
- Translation model learned from Hindi to Urdu without diacritics character alignment
- Translation model learned from Hindi to Urdu with diacritics cluster alignment
- Translation model learned from Hindi to Urdu without diacritics cluster alignment
- Translation model learned from Urdu with diacritics to Hindi character alignment
- Translation model learned from Urdu without diacritics to Hindi character alignment
- Translation model learned from Urdu with diacritics to Hindi cluster alignment
- Translation model learned from Urdu without diacritics to Hindi cluster alignment

For developing these translation models, we used the training script 'train-factored-phrase-model.perl' with options 'grow-diag-final' and 'msd-bidirectional-fe' (default options) for alignment and re-ordering respectively to learn translation models from different type of alignments. We also used different target language models or re-ordering models that we describe in the next section. In total, we have developed 24 SMT systems by combining different translation models and target language models.

3.1.4. Target language models

A *target language model* $\mathrm{P}(e)$ is a probabilistic model that scores the well-formedness of different translation solutions produced by the translation model [5, 106, 142, 193]. It generates a probability distribution over possible sequences of words and computes the probability of producing a given word w_1 given all the words that precede it in the sentence [5]. We have also developed different target language models for our different translation models depending on the alignment used in the translation model and the target language.

We broadly categorize them into *word language models* and *sentence language models*, discussed below.

3.1.4.1. Word language models

A *word language model* is a 6-gram statistical model that gives a probability distribution over possible sequences of characters and computes the probability of producing a given character or cluster c_1, given the 5 characters or clusters that precede it in the word. We developed 50,000 Hindi and Urdu (with and without diacritics) word lists for learning the Hindi-Urdu alignment and translation models. We then used the target side word lists to generate word language models.

For example, we have two types of *translation models* for Hindi to Urdu, one learned from *Hindi Urdu character alignments* and the other learned from *Hindi Urdu cluster alignments*. For each translation model, we developed a word language model depending on either character level word list or cluster level word list. More precisely, we developed *Urdu Word Language Models with Diacritics* (UWLMWD) from our character and cluster level Urdu word lists with diacritics using the SRILM freeware[38], and used them as target language models in the corresponding scriptural SMT systems. Similarly, we developed *Hindi Word Language Models* (HWLM) from our character and cluster level Hindi word lists and *Urdu Word Language Models without Diacritics* (UWLMWOD) from our Urdu (without diacritics) word lists.

3.1.4.2. Sentence language models

Similar to a word language model, a sentence language model is also a 6-gram statistical model that computes the probability of producing a given character or cluster c_1, given the 5 characters or clusters that precede it in the sentence. The Hindi Urdu pair is an under-resourced pair, but fortunately we were able to find monolingual corpora for Hindi and Urdu.

For Hindi, a Hindi corpus of more than 3 million words is freely made available by the "Resource Center for Indian Language Technology Solutions" of the Indian Institute of Technology Bombay (IITB)[39]. We processed this Hindi corpus and extracted a Hindi sentence corpus that contains one sentence per line. It has a total of 173,087 Hindi sentences. From this processed Hindi corpus, we developed a character-level Hindi corpus by introducing a space after each character. We also developed another clustered Hindi corpus applying our Hindi clustering finite-state transducer on the character-level Hindi corpus. For these two characters and cluster-

[38] http://www.speech.sri.com/projects/srilm/

[39] http://www.cfilt.iitb.ac.in/

level Hindi corpora, we developed character-level and cluster-level Hindi Sentence Language Models (HSLM) using the SRILM toolkit.

We were also able to collect a monolingual Urdu corpus (Reference # ELRA-W0037) of more than 2 million words from the "Evaluations and Language Resources Distribution Agency" (ELRD)[40]. This corpus was developed under the EMILLE[41] project of Lancaster University, UK. Like for the Hindi corpus, we processed this Urdu corpus and extracted from it a sentence corpus. It contains a total of 127,685 sentences. We developed a character-level and cluster-level Urdu corpus by introducing a space after each character and then by applying clustering. Finally, we developed character-level and cluster-level Urdu Sentence Language Models (USLM) using the SRILM toolkit.

3.1.5. SMT systems for Hindi-Urdu scriptural translation

Generally, an SMT system consists of a *translation model* $P(f|e)$ and a *target language el* $P(e)$. We developed 8 translation models and 18 target language models to build our Hindi-Urdu scriptural translation systems. By combining different translation and target language models, we have developed in total 24 SMT systems for the Hindi-Urdu scriptural translation.

12 SMT systems for Hindi to Urdu scriptural translation are shown in Figure 30.

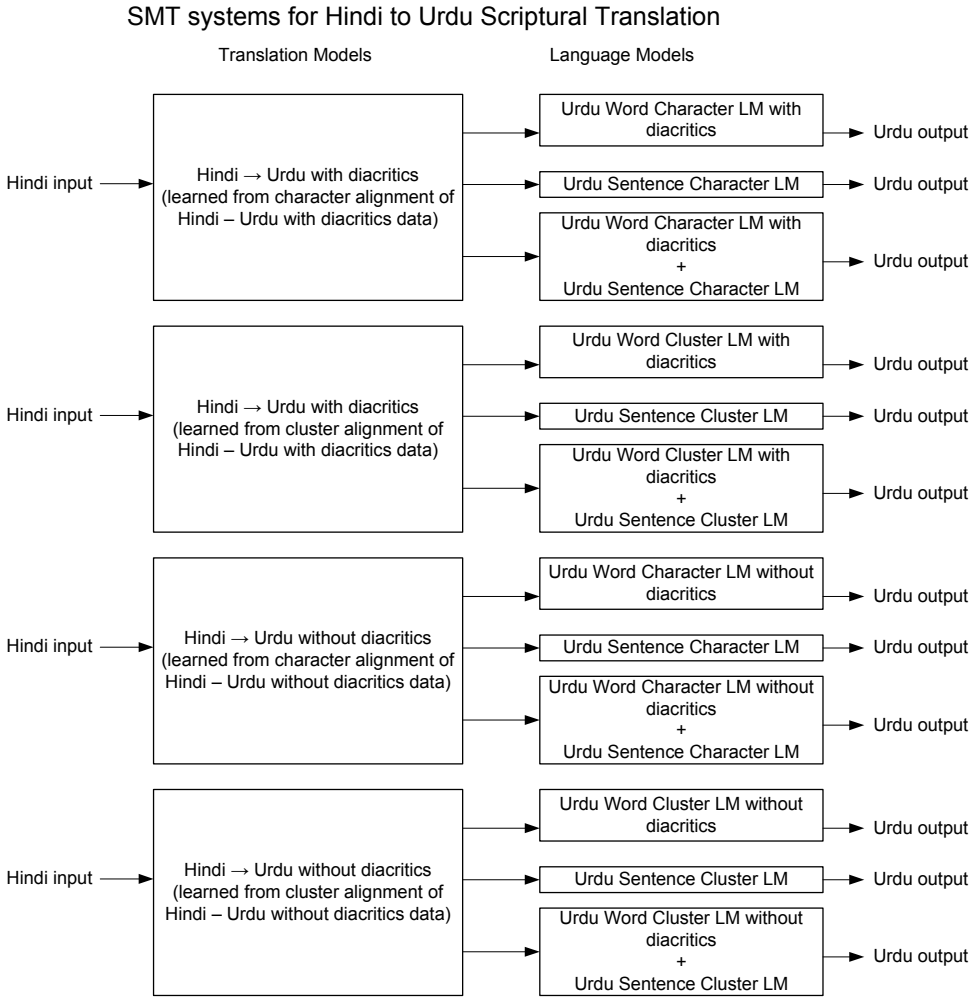


Figure 30. SMT Systems for Hindi to Urdu scriptural translation

[40] http://www.elda.org/

[41] http://www.emille.lancs.ac.uk/index.php

We built 4 translation models based on different Hindi-Urdu alignments and 8 target language models, discussed above. In the Moses toolkit, we can direct the SMT system to use multiple target language models. Thus we built 4 other target language models by combining our *Urdu (with or without diacritics) word language models (character and cluster level)* and *Urdu sentence language models (character and cluster level)*.

We also built 12 SMT systems for Urdu to Hindi scriptural translation. We developed 4 translation models based on different Urdu-Hindi alignments, discussed above. We have also developed two Hindi target side language models, described before. As for the Urdu target side, we have combined the *Hindi word language model* and the *Hindi sentence language model* to build a combined target language model. Figure 31 shows different SMT systems developed for Urdu to Hindi scriptural translation.

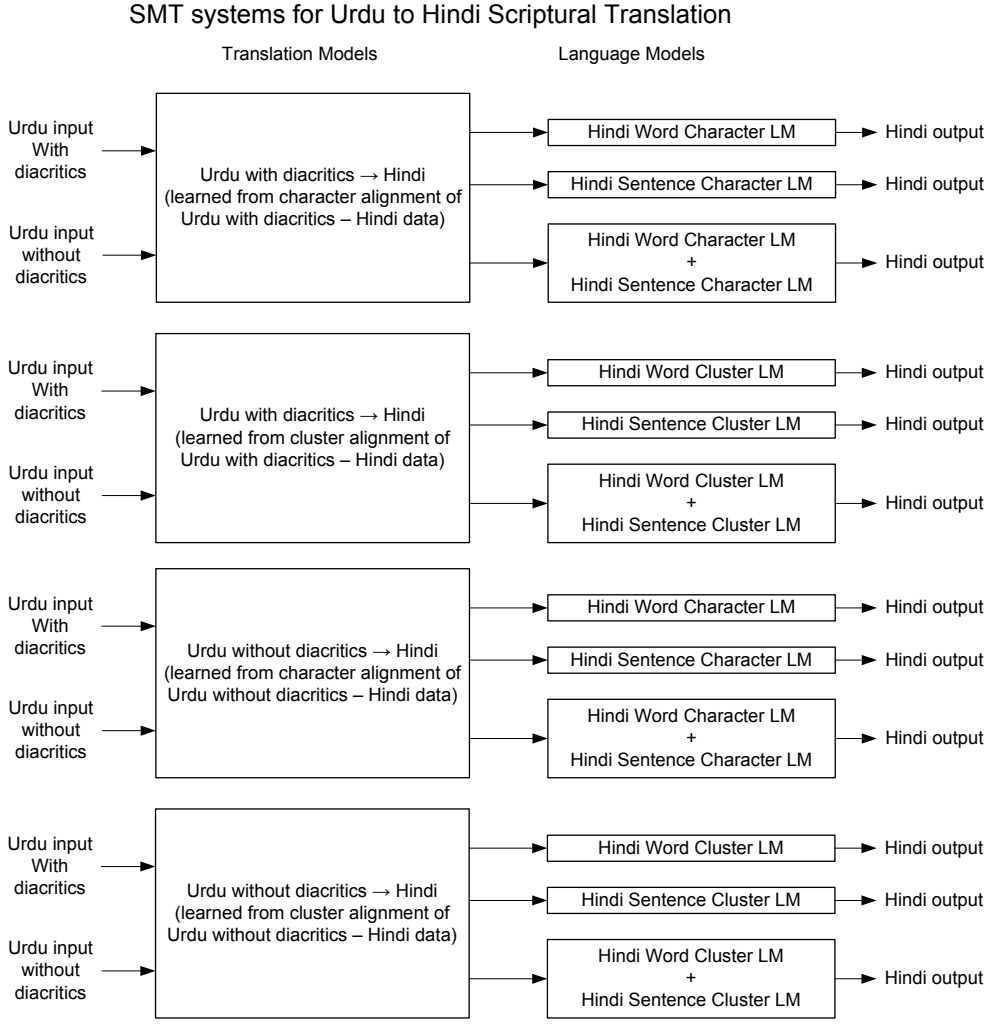


Figure 31. SMT systems for Urdu to Hindi scriptural translation

3.1.6. Experiments and results

As our translation models are learnt from parallel word lists, the input to these systems must be a word list and not a running text or a sentence. For this purpose, we first preprocess our Hindi or Urdu input text and generate a Hindi or Urdu word list before feeding it into the SMT system for translation. At the end, by post-processing the Urdu or Hindi word list produced by the SMT system, we generate the final Urdu or Hindi output text.

We have already developed the Hindi Urdu test sets, described in the previous chapter. We use the same test sets here for testing our SMT systems, except the HU Test Set 1, described later.

In this way, we can easily compare the results of different models developed for Hindi-Urdu scriptural translation.

3.1.6.1. Experiments

We used 50,000 Hindi – Urdu parallel words for learning the translation models. The other 2,753 were used as a reduced test set named HU Test Set 1. The remaining 2,500 Hindi – Urdu parallel words were used to tune the SMT systems. We tuned each SMT system based on character alignment or cluster alignment with the Moses 'mert-moses.pl' script. That doubled the number of our SMT systems, from 24 to 48.

During the application of these SMT systems on our test sets, we experimented with different parameters. For example, we applied our SMT systems on the input text with and without reordering to evaluate the effect of reordering and tuning on the performance of these SMT systems for Hindi-Urdu scriptural translation. This parameter selection again doubles the number of our experiments, from 48 to 96. By default, the Moses toolkit applies the reordering and we changed the default behavior of the Moses toolkit with the switch '-d 0' to restrict the reordering.

We have three different Hindi-Urdu test sets. For Hindi to Urdu, we performed 96 experiments for each Hindi input of each test set. Thus for testing Hindi to Urdu SMT systems, we performed 288 experiments. Each Urdu output is then post-processed, as described above.

We then computed character-level, word-level and sentence-level results. In total, we obtained 576 results for Hindi to Urdu scriptural translation. In the reverse direction, from Urdu to Hindi, the number of total experiments and results is also 576. These numbers are only for word accuracies. We have also computed NIST and BLEU scores for these experiments.

3.1.6.2. Results and discussion

We give results of particular interest for each test set, because it is not possible to give each and every result here and discuss it. We subdivide the results for each test set by the translation model, the alignment strategy, and the input/output type, because it is difficult to present all results in a one big table. First, we give and discuss the results of Hindi to Urdu scriptural translation for all three test sets. Then, we give the results for Urdu to Hindi translation. We give *sentence accuracy*, *word accuracy* and *character accuracy*.

3.1.6.2.1. Hindi to Urdu results

For Hindi to Urdu scriptural translation, Table 42 shows all the results of SMT systems for Hindi to Urdu scriptural translation, developed from Hindi-Urdu with diacritics (Hi-UrWD) and Hindi-Urdu without diacritics (Hi-UrWOD) *character alignments*, for HU Test Set 2.

The best results for the SMT systems, developed from Hi-UrWD character alignments are 71.5% and 5.5% at the word-level and the sentence level respectively. These results are shown in bold.

The best result at word-level is produced by the SMT system Hi-UrWD-USLM+ UWLMWD-No-Reordering. The same SMT system also produces the best result of 94.5% accuracy at character-level. On the other hand, it produces the second best result (4.5%) at sentence-level. According to the scales of Figure 27 and Figure 28, this system is classified as 'GOOD ENOUGH' and 'NULL', respectively.

The best result at sentence level is produced by the SMT system Hi-UrWD-USLM-No-Reordering and it is classified as 'AVERAGE' and 'OK' by the scales of Figure 27 and Figure 28 respectively.

The worst results for the same set of SMT systems are 26.1% (NULL) and 0.5% (NULL) at word-level and sentence level respectively.

The best results for the SMT systems, developed from Hi-UrWOD character alignments are 68.5% and 5.5% at word-level and sentence-level, respectively, shown in bold the second grind of the table below. In this case also, the best results at word-level and sentence-level are produced by different SMT systems.

SMT Model	Sentence Accuracy		Word accuracy		Character accuracy	
	default output	Processed output	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering	0.5%	3%	26.1%	65.7%	89.1%	93.3%
Hi-UrWD-UWLMWD-No-Reordering	0.5%	3%	26.1%	65.7%	89.1%	93.3%
Hi-UrWD-UWLMWD-Tuned-With-Reordering	1%	3%	34.4%	62.9%	88.7%	92.7%
Hi-UrWD-UWLMWD-Tuned-No-Reordering	1%	3%	34.4%	62.9%	88.7%	92.7%
Hi-UrWD-USLM-With-Reordering	1%	4%	48.9%	62.2%	84.9%	92.2%
Hi-UrWD-USLM-No-Reordering	1%	**5.5%**	49.7%	64.2%	85.8%	93.3%
Hi-UrWD-USLM-Tuned-With-Reordering	0.5%	3.5%	34.2%	63.3%	88.6%	92.6%
Hi-UrWD-USLM-Tuned-No-Reordering	0.5%	3.5%	34.2%	63.3%	88.6%	92.6%
Hi-UrWD-USLM+UWLMWD-With-Reordering	1%	4.5%	50.5%	70.9%	89.0%	94.3%
Hi-UrWD-USLM+UWLMWD -No-Reordering	1%	4.5%	50.8%	**71.5%**	89.2%	**94.5%**
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering	1%	3%	33.9%	62.7%	88.6%	92.6%
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering	1%	3%	33.9%	62.7%	88.6%	92.6%
Hi-UrWOD-UWLMWOD-With-Reordering	3%	3%	63.6%	63.6%	93.3%	93.3%
HiWOD-Ur-UWLMWOD-No-Reordering	3%	3%	63.6%	63.6%	93.3%	93.3%
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering	3%	3%	64.8%	64.8%	92.6%	92.6%
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering	3%	3%	64.8%	64.8%	92.6%	92.6%
Hi-UrWOD-USLM-With-Reordering	**5.5%**	**5.5%**	63.2%	63.2%	92.9%	92.9%
Hi-UrWOD-USLM-No-Reordering	**5.5%**	**5.5%**	63.8%	63.8%	93.5%	93.5%
Hi-UrWOD-USLM-Tuned-With-Reordering	3.5%	3.5%	64.8%	64.8%	92.8%	92.8%
Hi-UrWOD-USLM-Tuned-No-Reordering	3.5%	3.5%	64.8%	64.8%	92.8%	92.8%
Hi-UrWOD-USLM+UWLMWOD-With-Reordering	5%	5%	**68.5%**	**68.5%**	**93.7%**	**93.7%**
Hi-UrWOD-USLM+UWLMWOD -No-Reordering	5%	5%	**68.5%**	**68.5%**	**93.7%**	**93.7%**
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering	3%	3%	64.8%	64.8%	92.7%	92.7%
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering	3%	3%	64.8%	64.8%	92.7%	92.7%

Table 42. HU Test Set 2 results of Hindi to Urdu SMT systems (character alignments)

The word-level best result is produced by the SMT systems Hi-UrWOD-USLM+UWLMWOD-With-Reordering and Hi-UrWOD-USLM+UWLMWOD-No-Reordering. The same SMT systems also produce the best results at character-level. These two SMT systems are classified as 'AVERAGE' and 'OK' by the scales of Figure 27 and Figure 28, respectively.

The SMT systems that produce the best results at sentence-level are also classified as 'AVERAGE' and 'OK' by the scales of Figure 27 and Figure 28, respectively.

During out discussion of the results of our finite-state scriptural translation systems for the Indo-Pak languages, we have shown a sample Hindi input text in Figure 26. For the same sample input Hindi text, Table 43 shows the Urdu output of the SMT systems Hi-UrWD-USLM+UWLMWD-No-Reordering and Hi-UrWD-USLM-No-Reordering that produced the best results at word-level and sentence-level, respectively.

Urdu reference without diacritics	**Processed Urdu output of Hi-UrWD-USLM+UWLMWD-No-Reordering**
بھارتیہ سانسکرتک پرمپراؤں کپلا واتسیائن کلاؤں کے کشیتر میں سرجناتمک کاریہ پرشن بہن جی ، آپ کا ودیادھیین پرتیبھاسوچک رہے ہے اور اس کے بعد بھارت سرکار کے سنسکرتوبھاگ کے ادھیکش کے ناتے بھی آپ کا کاریہ پرچلوٹی کا رہے ہے ؛ اس پد پر کاریہ کرتے ہوئے آپ پر بھارت کے اترراشٹریہ سانسکرتک سمبندھوں اور بھارت کے بھیتر سانسکرتک متی کے نرمان کا دایتو رہے ہے ، اس کشیتر میں پرسی بھندبھن چیزیں آتی ہیں جیسے سنگرہالیہ ، پرامنو ، پرالیکھ ، گرنتھالی اور کلاؤں ؛ کیندریہ ستر پر بھی اور راجی ستر پر بھی ۔ آپ کا سرکاری کاریہ تو بھارت اور ودیشوں میں سوویدت رہے ہے پر بھارت کی پردرشنکاری اور سکھٹہ پلاسٹک کلاؤں کی ورگیکرن اور ویاکھیا کرنی کی کشیتر میں آپ کی سرجناتمک یوگدان کی باری میں کم سی کم اوست ستر کی ششٹ سامانیہ ویکتی کو کچھ ادھک جانکاری نہیں ہے ۔ کیا آپ ہماری پاٹھکوں کی لی بھارتیہ کلاؤں اور سوندریشاستر کی کشیتر میں کیے گیے نئی سرجناتمک اور انوپشاتمک کاریہ کا سنکشپت ولورن دینی کی کرپا کریں گی ؟	بھارتیہی سانسکرتک پرمپراہیں کپلا واتسیائن کلاؤں کی کشیتر میں سرجناتمک کاریہ پرشن بہن جی ، آپ کا ودیادھیین پرتیبھاسوچک رہے اور اس کی باد بھارت سرکار کی سنسکرتیوبھاگ کی ادھیکش کی ناتی بھی آپ کا کاریہ پرچلوٹی کا رہے ہے ؛ اس پد پر کاریہ کرتی ہوای آپ پر بھارت کی اترراشٹریی سانسکرتک سمبندھوں اور بھارت کی بھیتر سانسکرتک متی کی نرمان کا دایتو رہے ، اس کشیتر میں پسی بھندبھن چیزیں آتی ہیں جیسی سنگرہالی ، پرامنو ، پرالیکھ ، گرنتھالی اور کلاءں ؛ کیندریہ ستر پر بھی اور راجی ستر پر بھی ۔ آپ کا سرکاری کاریہ تو بھارت اور ودیشوں میں سویدت رہے ہے پر بھارت کی پردرشنکاری اور سکھٹی پلاسٹک کلاؤں کی ورگیکرن اور ویاکھیا کرنی کی کشیتر میں آپ کی سرجناتمک یوگدان کی باری میں کم سی کم اوست ستر کی ششٹ سامانی ویکتی کو کچھ ادھک جانکاری نہیں ہے ۔ کیا آپ ہماری پاٹھکوں کی لی بھارتیہی کلاؤں اور سوندریشاستر کی کشیتر میں کیے گیے نئی سرجناتمک اور انوپشاتمک کاریہ کا سنکشپت ولورن دینی کی کرپا کریں گی ؟

Urdu reference without diacritics	**Processed Urdu output of Hi-UrWD-USLM-No-Reordering**
بھارتیہ سانسکرتک پرمپراہیں کپلا واتسیائن کلاؤں کے کشیتر میں سرجناتمک کاریہ پرشن بہن جی ، آپ کا ودیادھیین پرتیبھاسوچک رہے ہے اور اس کے بعد بھارت سرکار کے سنسکرتوبھاگ کے ادھیکش کے ناتے بھی آپ کا کاریہ پرچلوٹی کا رہے ہے ؛ اس پد پر کاریہ کرتے ہوئے آپ پر بھارت کے اترراشٹریہ سانسکرتک سمبندھوں اور بھارت کے بھیتر سانسکرتک متی کے نرمان کا دایتو رہے ہے ، اس کشیتر میں پسی بھندبھن چیزیں آتی ہیں جیسے سنگرہالیہ ، پرامنو ، پرالیکھ ، گرنتھالیہ اور کلاءں ؛ کیندریہ سطر پر بھی اور راجیہ سطر پر بھی ۔ آپ کا سرکاری کاریہ تو بھارت اور ودیشوں میں سودت رہے ہے پر بھارت کی پردرشنکاری اور سکھٹہ پلاسٹک کلاءوں کے ورگیکرن اور ویاکھیا کرنے کے کشیتر میں آپ کے سرجناتمک یوگدان کے بارے میں کم سے کم اوسط سطر کے ششٹ سامانیہ ویکتی کو کچھ ادھک جانکاری نہیں ہے ۔ کیا آپ ہمارے پاٹھکوں کے لیے بھارتیہ کلاءوں اور سوندریشاستر کے کشیتر میں کیے گئی نئے سرجناتمک اور انوپشاتمک کاریہ کا سنکشپت دوران دینے کی کرپا کریں گی ؟	بھارتیہی سنسکرتیک پرمپراہیں کپیلا واتسیائن کلاؤں کی کھیتر میں سرجناتمک کاریہ پرشن بہن جی ، آپ کا ودیادھیین پرطبھاسوچکا رہا ہی اور اس کی باد بھارت سرکار کی سنسکرتیوبھاگا کی ادھیکس کی ناتی بھی آپ کا کاریہ پرچلوٹی کا رہا ہی ؛ اس پد پر کاریہ کرتی ہوای آپ پر بھارت کی اترراشٹریے سنسکرتیک سمبندھوں اور بھارت کی کھیتر سنسکرتیک متی کی نرمان کا دایتو رہا ہی ، اس کھیتر میں پسی کھیناکھینین چیزیں آتی ہیں جیسے سنگرہالی ، پرامنو ، پرالیکھ ، گرنتھالی اور کلایں ؛ کیندریہ سر پر بھی اور راجی سر پر بھی ۔ آپ کا سرکاری سکاریہ تو بھارت اور ودیشوں میں سویدیت رہا ہی پر بھارت کی پردرشنکاری اور سکھٹی پلاسٹک کلاؤں کی ورگیکرن اور ویاکھایا کرنے کی کھیتر میں آپ کی سرجناتمک یوگدان کی بہاری میں کم سی کم اوست سر کی شکسیت سامانی ویکتی کو کچھ ادھیک جانکاری نہیں ہی ۔ کیا آپ ہماری پاٹکوں کی لی بھارتیہی کلاؤں اور سوندریشاستر کی کھیتر میں کیے گیے نئے سرجناتمک اور انوے شناتمک کاریہ کا سنکشت دوران دینو کی کرپا کریں گی ؟

Table 43: Sample Hindi to Urdu outputs for SMT systems with the best results

On average, there are 9.4 and 16 errors per sentence in the Urdu outputs of the SMT systems Hi-UrWD-USLM+UWLMWD-No-Reordering and Hi-UrWD-USLM-No-Reordering. In terms of usability of the output text, these outputs are not usable and require a huge amount of effort for post-editing. Therefore, these SMT systems would be ranked quite low by a user.

The sentence level classification of a scriptural translation system is important because it correlates well with the user satisfaction and the usability of the scriptural translation output according to our observation of the results of the finite-state scriptural translation and the results reported above for the SMT systems for scriptural translation. Therefore, it is of our primary interest to improve the sentence level accuracy and classification.

From here onward, we will not give all the results like we did in Table 42. We report only the results of particular interest. The complete tables with all the results are given in Annex 7.

For HU Test Set 3, the best results produced by the SMT Hindi to Urdu scriptural translation systems, developed from Hindi-Urdu with diacritics (Hi-UrWD) and Hindi-Urdu without diacritics (Hi-UrWOD) *character alignments* are shown in Table 44.

SMT Model	Sentence Accuracy		Word accuracy		Character accuracy	
	default output	processed output	default output	processed output	default output	processed output
Hi-UrWD-USLM-No-Reordering	0.9%	**4%**	57.3%	**71.1%**	90.6%	93.3%

Table 44: HU Test Set 3 best results of Hindi to Urdu SMT systems (character alignments)

The SMT system of Table 44 is classified as 'GOOD ENOUGH' and 'NULL' according to the scales of Figure 27 and Figure 28, respectively.

For HU Test Set 2 and 3, the best results produced by the SMT systems, developed from the Hi-UrWD and Hi-UrWOD cluster alignments, are given in Table 45.

HU Test Set 2

SMT Model	Sentence Accuracy		Word accuracy		Character accuracy	
	default output	processed output	default output	processed output	default output	processed output
Hi-UrWD-USLM-No-Reordering	1%	**5.5%**	53.4%	66.6**%**	86.2%	93.6%
Hi-UrWOD-USLM-With-Reordering	**5.5%**	**5.5%**	65.3%	65.3%	93.0%	93.0%
Hi-UrWOD-USLM-No-Reordering	**5.5%**	**5.5%**	66.2%	66.2%	93.6%	93.6%
Hi-UrWOD-USLM+UWLMWOD-Tuned-With-Reordering	**5.5%**	**5.5%**	69.5%	69.5%	93.6%	93.6%
Hi-UrWOD-USLM+UWLMWOD-Tuned-No-Reordering	**5.5%**	**5.5%**	**69.7%**	**69.7%**	93.6%	93.6%

HU Test Set 3

SMT Model	Sentence Accuracy		Word accuracy		Character accuracy	
	default output	processed output	default output	processed output	default output	processed output
Hi-UrWD-USLM-No-Reordering	0.9%	**4.9%**	58.0%	69.3%	89.0%	93.4%
Hi-UrWOD-USLM+UWLMWOD-Tuned-With-Reordering	3.5%	3.5**%**	**68.0%**	**68.0%**	92.7%	92.7%
Hi-UrWOD-USLM+UWLMWOD-Tuned-No-Reordering	3.5%	3.5**%**	**68.0%**	**68.0%**	92.7%	92.7%

Table 45: HU Test Set 2 and 3 best results of Hindi to Urdu SMT systems (cluster alignments)

For HU Test Set 2, the SMT system producing the best results is classified as 'AVERAGE' and 'OK' according to the classification scales of Figure 27 and Figure 28, respectively. For HU Test Set 3, the sentence-level classification the SMT system is 'NULL' and the word-level classification of the SMT system is also 'AVERAGE'.

For HU Test Set 1, the best results are 78.4% and 79.7% for the default and the processed Urdu output by the SMT systems Hi-UrWOD-USLM+UWLMWOD-Tuned-No-Reordering and Hi-UrWD-USLM+UWLMWD-Tuned-No-Reordering, developed from the *cluster alignments*, respectively. HU Test Set 1 consists of a word list, so there is no meaning of sentence-level results here. These SMT systems are classified as 'GOOD ENOUGH' according to the classification scale of Figure 27.

For HU Test Set 1, the best results are 78.3% and 80.2% for the default and the processed Urdu output by the SMT systems Hi-UrWOD-USLM+UWLMWOD-Tuned-No-Reordering and Hi-UrWD-USLM+UWLMWD-Tuned-No-Reordering, developed from the *character alignments*, respectively. HU Test Set 1 consists of a word list, so there is no meaning of sentence-level results here. These SMT systems are classified as 'GOOD ENOUGH' and 'GOOD' according to the classification scale of Figure 27.

We have computed the edit distances, BLEU and NIST scores that are given in Annex 7.

3.1.6.2.2. Urdu to Hindi results

Table 46 shows the best results of Urdu to Hindi scriptural translation by our SMT systems, developed from the Urdu with diacritics to Hindi (UrWD-Hi) and Urdu without diacritics to Hindi (UrWOD-Hi) character alignments for HU Test Set 2 and 3.

HU Test Set 2

SMT Model	Sentence Accuracy		Word accuracy		Character accuracy	
	with diacritics	without diacritics	with diacritics	without diacritics	with diacritics	without diacritics
UrWD-Hi-HSLM+HWLM-Tuned-With-Reordering	**5.5%**	2%	**72.2%**	57.9**%**	91.8%	85.8%
UrWD-Hi-HSLM+HWLM-Tuned-No-Reordering	**5.5%**	2%	**72.2%**	57.9**%**	91.8%	85.8%
UrWOD-Hi-HSLM+HWLM-With-Reordering	0.5%	5%	50.1%	**77.0%**	86.8%	94.6%
UrWOD-Hi-HSLM+HWLM-No-Reordering	0.5%	5%	50.1%	**77.0%**	85.3%	94.6%

HU Test Set 3

SMT Model	Sentence Accuracy		Word accuracy		Character accuracy	
	with diacritics	without diacritics	with diacritics	without diacritics	with diacritics	without diacritics
UrWD-Hi-HSLM+HWLM-Tuned-With-Reordering	5.3%	0.4%	77.8%	57.9%	94.4%	86.7%
UrWD-Hi-HSLM+HWLM-Tuned-No-Reordering	5.3%	0.4%	77.8%	57.9%	94.4%	86.7%
UrWOD-Hi-HSLM+HWLM-With-Reordering	0%	0.4%	44.8%	**60.1%**	87.3%	90.8%
UrWOD-Hi-HSLM+HWLM-No-Reordering	0%	0.4%	44.8%	**60.1%**	87.7%	90.8%

Table 46: HU Test Set 2 and 3 best results of Urdu to Hindi SMT systems (character alignments)

We have previously given the sample Urdu input texts with and without diacritics from HU Test Set 2 in Table 25 and Table 26, respectively (see page 42). In Table 47, we give the Hindi output for the sample Urdu input text with diacritics (Urdu input text of Table 25) of the SMT system UrWD-Hi-HSLM+HWLM-Tuned-No-Reordering with its Hindi reference.

On average, the Hindi output of Table 47 contains 10.8 errors per sentence. The SMT system is classified as 'OK' according to the sentence-level classification scale of Figure 28. A real user of the system would rate this output very low or even totally unacceptable.

Table 48 shows the Hindi output of the SMT system UrWOD-Hi-HSLM+HWLM-No-Reordering for the sample Urdu input without diacritics (Urdu input text of Table 26). The SMT system is also classified as 'OK' according to the classification scale of Figure 28. The Hindi output of Table 48 also contains 10.8 errors per sentence.

In terms of usability and user satisfaction, these results are not very good. Rather, they are very bad. If we build an online service with the SMT system, then it would be possible that a user would try a few times to translate his text. But he would eventually stop using this online service due to very poor results in terms of post-editing efforts and user satisfaction.

Hindi reference	Hindi output from the Urdu text with diacritics UrWD-Hi-HSLM+HWLM-Tuned-No-Reordering
भारतीय सांस्कृतिक परंपराएँ कपिला वात्स्यायन कलाओं के क्षेत्र में सृजनात्मक कार्य प्रश्न बहन जी , आप का विद्याध्ययन प्रतिभासूचक रहा है और उस के बाद भारत सरकार के संस्कृतिविभाग के अध्यक्ष के नाते भी आप का कार्य उच्चकोटि का रहा है ; इस पद पर कार्य करते हुए आप पर भारत के अन्तर्राष्ट्रीय सांस्कृतिक सम्बन्धों और भारत के भीतर सांस्कृतिक नीति के निर्माण का दायित्व रहा है , इस क्षेत्र में ऐसी भिन्नभिन्न चीज़ें आती हैं जैसे संग्रहालय , पुरातत्व , पुरालेख , ग्रंथालय और कलाएँ ; केन्द्रीय स्तर पर भी और राज्य स्तर पर भी । आप का सरकारी कार्य तो भारत और विदेशों में सुविदित रहा है पर भारत की प्रदर्शनकारी और सुघट्य प्लास्टिक कलाओं के वर्गीकरण और व्याख्या करने के क्षेत्र में आप के सृजनात्मक योगदान के बारे में कम से कम औसत स्तर के शिक्षित सामान्य व्यक्ति को कुछ अधिक जानकारी नहीं है । क्या आप हमारे पाठकों के लिए भारतीय कलाओं और सौन्दर्यशास्त्र के क्षेत्र में किये गये अपने सृजनात्मक और अन्वेषणात्मक कार्य का संक्षिप्त विवरण देने की कृपा करें गी ?	भारतीय सांसकृतिक परंपराईं कपिला वातसयायन कलावं के क्ंषेतर में सरज्ञातमक कार्य परेषन बहन जी , आप का विदयाधयेन परतिभासूचक रहा है और उस के बअद भारत सरकार के संसकृतिविभाग के अधेकष के नाते भी आप का कार्य उच्चकोटी का रहा है ; इस पद पर कार्य करते हूए आप पर भारत के अनतर्राषटरीया सांसकृतिक समबनधों और भारत के भीतर सांसकृतिक नीति के निरमान का दायितो रहा है , इस क्ंषेतर में ऐसी भिन्नभिन्न चीज़ें आती हैं जैसे संग्राल्य , पुराततो , पुरालेख , गरंथाल्य और कलाईं ; केनदरीया सतर पर भी और राज्य सतर पर भी । आप का सरकारी कार्य तो भारत और विदेशों में सुविदित रहा है पर भारत की परदरशंकारी और सुघट्य पलासटिक कलावं के वरगीकरण और वयाखया करने के क्ंषेतर में आप के सरज्ञातमक योगदान के बारे में कम से कम औसत सतर के षिकषित सामान्य वेकती को कुछ अधिक जानकारी नहीं है । कया आप हमारे पाठकों के लिये भारतीय कलावं और सौनदरेषासतर के क्ंषेतर में किये गढे अपने सरज्ञातमक और अनवेशनातमक कार्य का सनकषिपत विवरन देने की कृपा करें गी ?

Table 47: A sample SMT system Hindi output with its reference

Hindi reference	Hindi output from the Urdu text with diacritics UrWD-Hi-HSLM+HWLM-Tuned-No-Reordering
भारतीय सांस्कृतिक परंपराएँ कपिला वात्स्यायन कलाओं के क्षेत्र में सृजनात्मक कार्य प्रश्न बहन जी , आप का विद्याध्ययन प्रतिभासूचक रहा है और उस के बाद भारत सरकार के संस्कृतिविभाग के अध्यक्ष के नाते भी आप का कार्य उच्चकोटि का रहा है ; इस पद पर कार्य करते हुए आप पर भारत के अन्तर्राष्ट्रीय सांस्कृतिक सम्बन्धों और भारत के भीतर सांस्कृतिक नीति के निर्माण का दायित्व रहा है , इस क्षेत्र में ऐसी भिन्नभिन्न चीज़ें आती हैं जैसे संग्रहालय , पुरातत्व , पुरालेख , ग्रंथालय और कलाएँ ; केन्द्रीय स्तर पर भी और राज्य स्तर पर भी । आप का सरकारी कार्य तो भारत और विदेशों में सुविदित रहा है पर भारत की प्रदर्शनकारी और सुघट्य प्लास्टिक कलाओं के वर्गीकरण और व्याख्या करने के क्षेत्र में आप के सृजनात्मक योगदान के बारे में कम से कम औसत स्तर के शिक्षित सामान्य व्यक्ति को कुछ अधिक जानकारी नहीं है । क्या आप हमारे पाठकों के लिए भारतीय कलाओं और सौन्दर्यशास्त्र के क्षेत्र में किये गये अपने सृजनात्मक और अन्वेषणात्मक कार्य का संक्षिप्त विवरण देने की कृपा करें गी ?	भारतीय सांसकृतिक परंपराईं कपिला वातसयायन कलावं के क्ंषेतर में सरज्ञातमक कार्य परेषन बहन जी , आप का विदयाधयेन परतिभासूचक रहा है और उस के बअद भारत सरकार के संसकृतिविभाग के अधेकष के नाते भी आप का कार्य उच्चकोटी का रहा है ; इस पद पर कार्य करते हूए आप पर भारत के अनतर्राषटरीया सांसकृतिक समबनधों और भारत के भीतर सांसकृतिक नीति के निरमान का दायितो रहा है , इस क्ंषेतर में ऐसी भिन्नभिन्न चीज़ें आती हैं जैसे संग्राल्य , पुराततो , पुरालेख , गरंथाल्य और कलाईं ; केनदरीया सतर पर भी और राज्य सतर पर भी । आप का सरकारी कार्य तो भारत और विदेशों में सुविदित रहा है पर भारत की परदरशंकारी और सुघट्य पलासटिक कलावं के वरगीकरण और वयाखया करने के क्ंषेतर में आप के सरज्ञातमक योगदान के बारे में कम से कम औसत सतर के षिकषित सामान्य वेकती को कुछ अधिक जानकारी नहीं है । कया आप हमारे पाठकों के लिये भारतीय कलावं और सौनदरेषासतर के क्ंषेतर में किये गढे अपने सरज्ञातमक और अनवेशनातमक कार्य का सनकषिपत विवरन देने की कृपा करें गी ?

Table 48: A sample SMT system Hindi output with its reference for Urdu input without diacritics

In general, the sentence-level accuracy of an SMT system for scriptural translation is always between 5% and 10%, so that it is always classified as 'OK' according to the scale of Figure 28. The reason behind this very low accuracy might be the training data. In our case, the training data is a parallel word list and not a parallel corpus (a usual case for a general SMT system). Unfortunately, we do not have any Hindi-Urdu parallel corpus to test our hypothesis that the

accuracy of Hindi-Urdu scriptural translation can be improved by training the SMT models with a Hindi-Urdu parallel corpus instead of using a Hindi-Urdu parallel word list.

In terms of usability, user satisfaction and effectiveness, the scriptural translation SMT models would be classified very low and bad by users in the real life. All the results of our SMT experiments are given in Annex 7.

3.1.6.3. Effects of different parameters on SMT results

In the words of [70], "It is an open question whether improved word alignment actually improves statistical MT". A *translation model* is a fundamental and crucial part of an SMT system and it is learnt from the word alignment. Thus our hypothesis is that one can build better SMT systems with better word alignments.

In our results, we observed that the improvement of the cluster alignment improves sometimes slightly and sometimes considerably the quality of the SMT systems, but it is not always true. Its effect on results of HU Test Set 1 from Hindi to Urdu scriptural translation systems is shown in Figure 32.

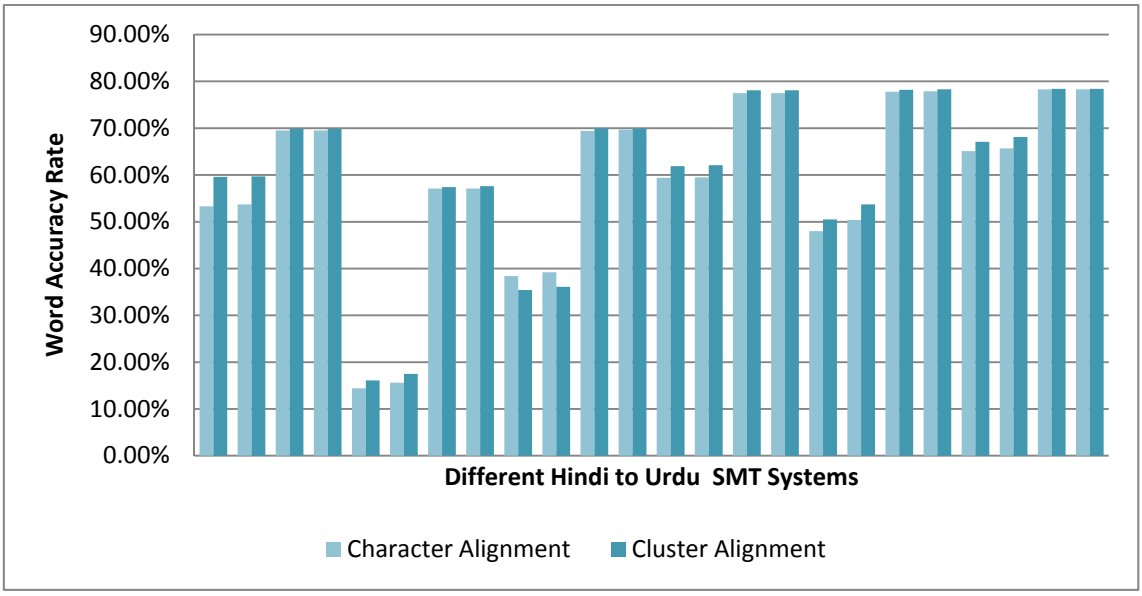


Figure 32. Effect of cluster alignments on SMT systems from Hindi to Urdu scriptural translation

The effect of cluster alignment on the quality of Hindi-Urdu SMT systems is shown in Figure 33 on results of HU Test Set 2.

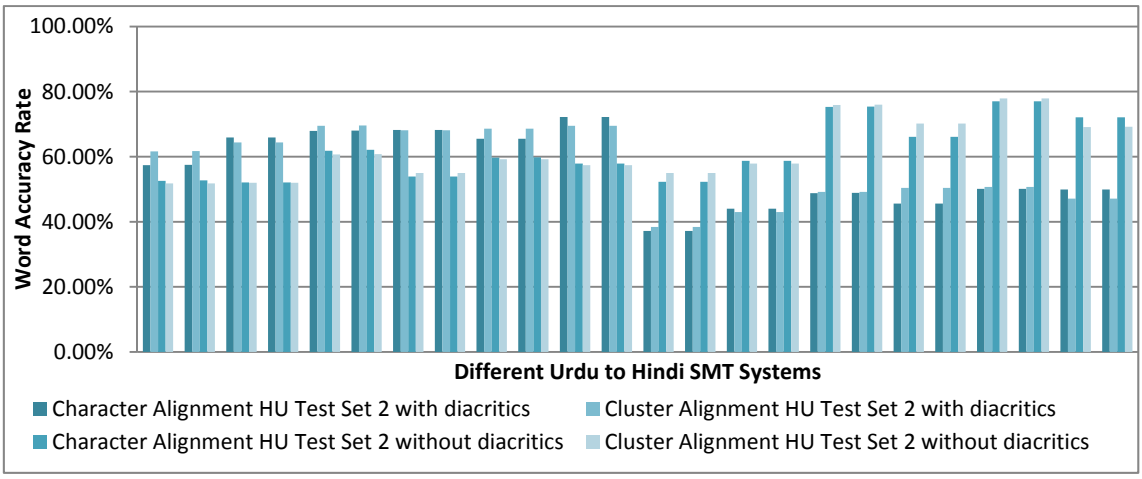


Figure 33. Effect of cluster alignments on SMT systems from Urdu to Hindi scriptural translation

In our results, cluster alignment shows a mixed behavior, as shown by the above figures. Thus we cannot conclude that improving the word alignment can help to develop an better quality SMT system.

From the output of each Hindi to Urdu SMT system, we developed another processed output by removing all the diacritical marks. This post-processing on the target side always increases the performance in terms of user acceptability because a native Urdu speaker is used to see Urdu text without diacritics. Most of the time, we get an increase of more than 20% in the performance of our systems. Sometimes, this gain is more than 30%. The effect of this post-processing on the results of SMT systems developed from Hindi Urdu with diacritics cluster alignments is shown in Figure 34.

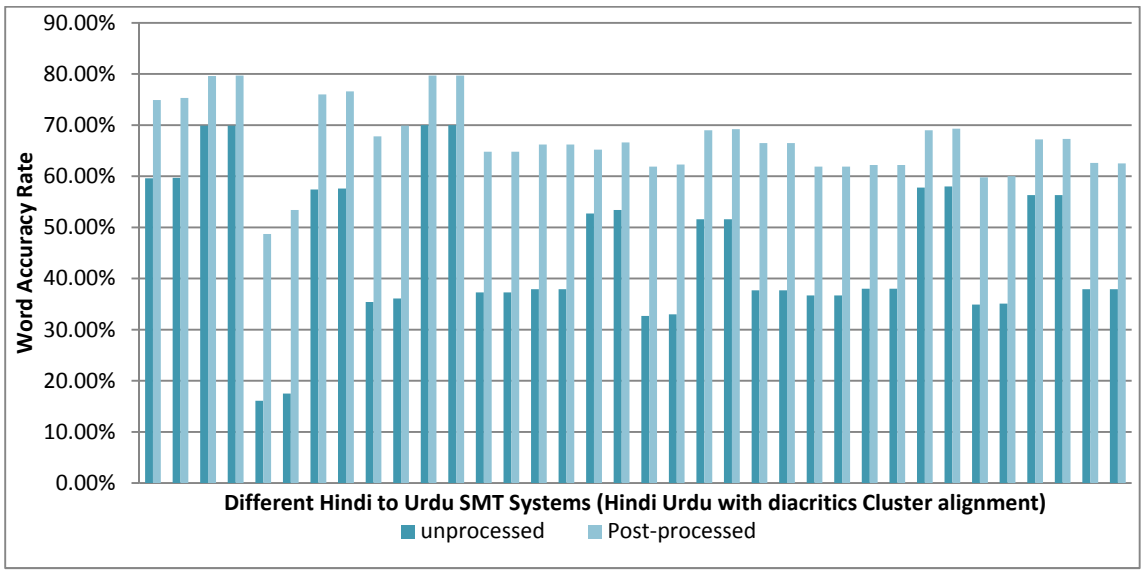


Figure 34. Effect of post-procsseding on Hindi ot Urdu SMT systems with cluster alignments

We varied the *reordering* parameter of the Moses decoder during the testing phase of our SMT systems. If we restrict or do not allow the reordering, then the word accuracy of our systems either increases or remains unaffected at all in both directions of Hindi Urdu scriptural translation.

The *tuning* process has an interesting effect on the performance of our scriptural translation systems. A tuned system increases the *word accuracy* when these systems are applied to the HU Test Set 1, the test data developed from the Urdu, Hindi and English dictionary data [151], while it has a mixed effect on the performance of our SMT systems. The tuning process gives us an increase of merely 1% to approximately 35% for HU Test Set 1. The effect of reordering and tuning on the performance of Urdu to Hindi SMT systems developed from Urdu without diacritics-Hindi character alignment is shown in Figure 35 on Urdu input without diacritics.

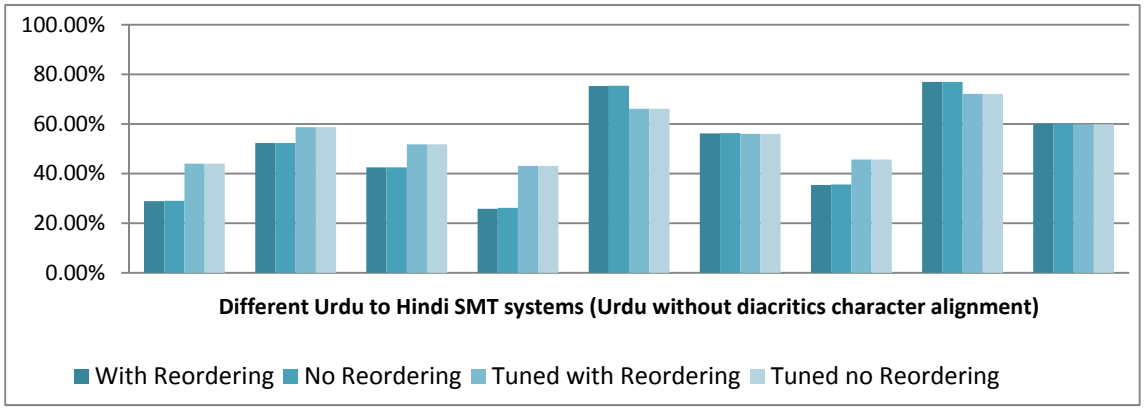


Figure 35. Effect of reordering and tunning on Urdu to Hindi scriptural translation

In the graph above, it is important to note that only two set of systems have crossed the 60% word accuracy mark. For Urdu to Hindi scriptural translation, we have reached the maximum 77.90% and 82.80% word accuracies for Urdu input without and with diacritics, respectively. For Hindi to Urdu, we have observed the maximum word accuracies of 78.40% and 80.20% for default and processed Urdu output, respectively.

3.2. Hybrid model for scriptural translation

The results of our scriptural SMT systems (less than 7% sentence accuracy in general) are not good in terms of user satisfaction, usability and effectiveness in real usage. On the other hand, the expert scriptural translation systems, implemented as finite-state transducers, produce better results than the SMT systems. The finite-state expert scriptural translation system translates the sample Hindi text of Figure 26 in Urdu with 9 errors (1.8 errors per sentence) while the scriptural SMT system translated it with 47 errors (9.4 errors per sentence).

The quality of the finite-state scriptural translation system is practical and it is usable in real life, but its quality is badly affected by the vocabulary diversity across languages or dialects and transliterational/transcriptional ambiguities. For example, our Hindi-Urdu finite-state scriptural translation system translates HU Test Set 3 (a test set of Urdu origin) with a sentence accuracy less than 10%. On the other hand, it translates HU Test Set 2 (a test set of Hindi origin) with a sentence accuracy of more than 25%. It is classified as 'OK' and 'GOOD' for HU Test Set 3 and HU Test Set 2 respectively. We already discussed the reasons of this variation in sentence accuracy in Section 2.4.2.

The missing information in the source text is another factor that appallingly affects the translation quality of the scriptural translation systems. The absence of diacritical marks decreases the sentence accuracy of the finite-state scriptural translation from 73.9% to 0.4% when it is applied to the Urdu input text of HU Test set 3.

Our expert systems (non-probabilistic finite-state transducers) for scriptural translation systems are not capable to handle vocabulary diversity, transliteration/transcriptional ambiguities and missing information. A scriptural translation expert system uses the character level knowledge to translate a source text in a target text. To handle the vocabulary diversity, transliterational/transcriptional ambiguities and missing information, it requires a word level knowledge.

We propose a hybrid model that is able to handle the problem of missing information, transliterational/transcriptional ambiguities and vocabulary diversity by using the target language models.

That model is a multilevel process. It firsts performs the scriptural translation of the source text in the target text and relates each source word with a set of target words. The cardinality of this set is 1 in most of the cases, but can reach a maximum of 4. The target language model is used as a *statistical filter* that uses the target language knowledge to filter out the correct solution.

3.2.1. Monolingual data

We have already described the Hindi corpus used while discussing the SMT systems for Hindi Urdu scriptural translation. That corpus contains more than 3 million words. We have used it for developing our Hindi word filter to filter out the correct vowelized Hindi word from a collection of candidate translations of an Urdu word.

3.2.2. Language models

We first extracted all 173,088 sentences from the Hindi corpus. We removed all punctuation marks. Then we added tags '<s>' and '</s>' at the start and at the end of each sentence. We

then trained a tri-gram Hindi language model using the command 'ngram-count' of the SRILM toolkit [175].

This Hindi language model serves the purpose of a statistical filter. This statistical filter is merely a weighted finite-state transducer that produces the best possible word in case of ambiguities or the unique solution for the input word. In the case of our example word shown above, there exists only one possible solution.

3.2.3. Word ambiguity network

We have created an ambiguity network for each unique word of our Hindi corpus accepting all Hindi words produceable by a naïve character-based transduction from the corresponding Urdu word. For example, the Urdu word اردو [ʊrd̤u] (Urdu) is converted into the Hindi word अरदो* (a wrong translation in Hindi) by Urd to Hindi finite-state scriptural translation system. Figure 36 shows the confusion network of all 42 possible words, with all possible vowels. Only one candidate उरदू [ʊrd̤u] is the correct Hindi word that we want to be filtered out sybsequently with the help of Hindi corpus. The Hindi corpus contains in total 120,538 unique words. We have extracted all these unique words.

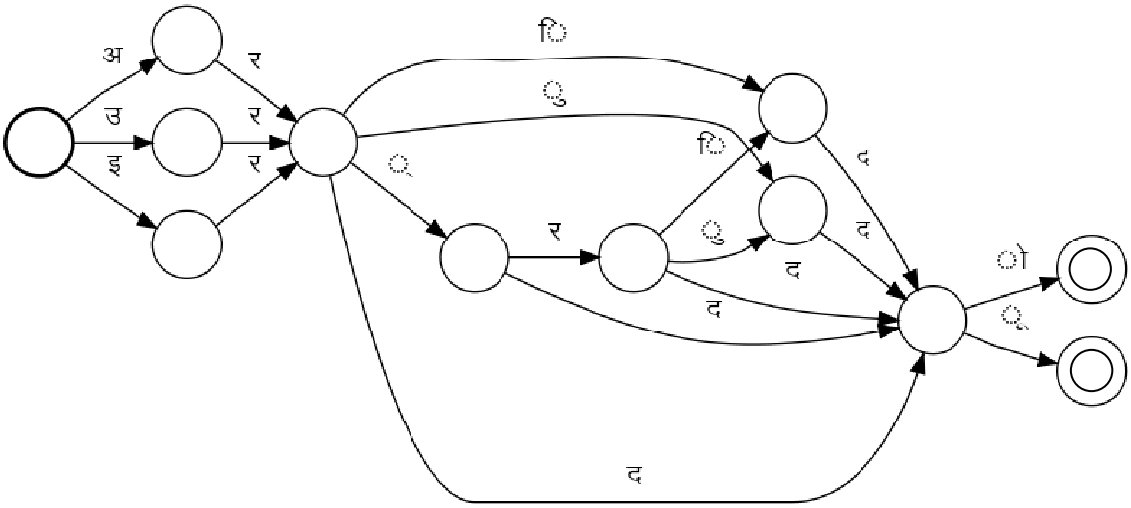


Figure 36. Word confusion network example

The process of development of a confusion network for each word present in the Hindi corpus is a regular process. Thus we have developed a finite-state transducer, say R (a regular relation). For a given Hindi word w, $R(w)$ is a confusion automaton. This transducer takes a Hindi word from the Hindi corpus and develops a list of all possible variations of that word by extracting all paths from the associated automaton. Figure 37 shows a few rules of this transducer. The complete transducer is given in Annex 6.

ि (→) NULL
ी (→) े
ु (→) NULL
ू (→) ो

Figure 37. Example rules for generating a word confusion network

All these rules are ambiguous. They mean that the character to the left of the (→) operator may be converted into to the right of the operator. Thus each rule doubles the number of possible variations for the input corpus Hindi word. Figure 38 shows the all possible variations of our example word of Figure 36.

अरदो	अरुदू	उरदो	उरुदू	इरदो	इरुदू
अरदू	अर्रदो	उरदू	उर्रदो	इरदू	इर्रदो
अर्दो	अर्रदू	उर्दो	उर्रदू	इर्दो	इर्रदू
अर्दू	अर्रिदो	उर्दू	उर्रिदो	इर्दू	इर्रिदो
अरिदो	अर्रिदू	उरिदो	उर्रिदू	इरिदो	इर्रिदू
अरिदू	अर्रुदो	उरिदू	उर्रुदो	इरिदू	इर्रुदो
अरुदो	अर्रुदू	उरुदो	उर्रुदू	इरुदो	इर्रुदू

Figure 38. Possible word variations for the example word

These confusion networks are deployed in the form of a *word map* file in which each line contains a possible variation of a Hindi corpus word followed by all those corpus words from which this possible variation can be generated by applying our finite-state transducer. Figure 39 shows some entries of our word map file.

उरदू	उर्दु उर्दू
उरदूको	उर्दूको
उरदूहनदी	उर्दूहिन्दी
उरदूहिनदी	उर्दूहिन्दी
टरोन	ट्रॉन ट्रोन
टली	टली टिली
टेकंअलोज़ी	टैक्नालोजी
टैक्नालॉजी	टैक्नालोजी
टले	टली टिली टुले
अंक	अंक इंक इनक
अंगरेज़	अँगरेज अँग्रेज अँग्रेज़ अंगरेज अंग्रेज अंग्रेज़
अंग़रेज़	अँगरेज अँग्रेज अँग्रेज़ अंगरेज अंग्रेज अंग्रेज़
टेकंअलोजी	टेकनालॉजी टेक्नालॉजी टेक्नालोजी टैकनालॉजी
टैक्नालॉजी	टेकनालॉजी टेक्नालॉजी टेक्नालोजी टैकनालॉजी
मने	मणि मनि मनी मने मिनि मिनी मुणि मुनि मुने
ढूंढ़ते	ढूँढती ढूँढ़ती ढूँढते ढूँढ़ते ढूंढती ढूंढ़ती ढूंढते ढूंढ़ते ढूंढ़ते
परे	परि परी परे पिरे पुरि पुरी पुरे प्री प्रे
बने	बणि बनि बनी बने बनै बिनि बिनी बिने बुनि बुनी बुने

Figure 39. A sample word map

In total, the word map file contains 962,893 entries. There are 49,378 entries with 2 words. The maximum number of possible words found for an entry word is 11. Table 49 shows the analysis of these ambiguities in the word map.

Ambiguities	**No. of entries**	**Ambiguities**	**No. of entries**
2	49,378	7	66
3	6,698	8	29
4	1811	9	6
5	481	10	0
6	182	11	1

Table 49. No. of Ambiguites in word map

We have already said in the previous chapter that the Hindi community commonly uses the characters क [k], ख [kʰ], ग [g], ज [ʤ], ड [ɖ], ढ [ɖʰ] and फ [p] instead of the characters क़ [q], ख़ [x], ग़ [ɣ], ज़ [z], ड़ [ɽ], ढ़ [ɽʰ] and फ़ [f] respectively due to their shape similarities. We have highlighted

a line in our sample word net which contains 6 ambiguities. If we standardize the Hindi corpus, then we will be end up with a unique corpus word for entries of that kind.

3.2.4. Hybrid Model for scriptural translation

Figure 40 shows the architecture of the proposed *hybrid model for scriptural translation* [120].

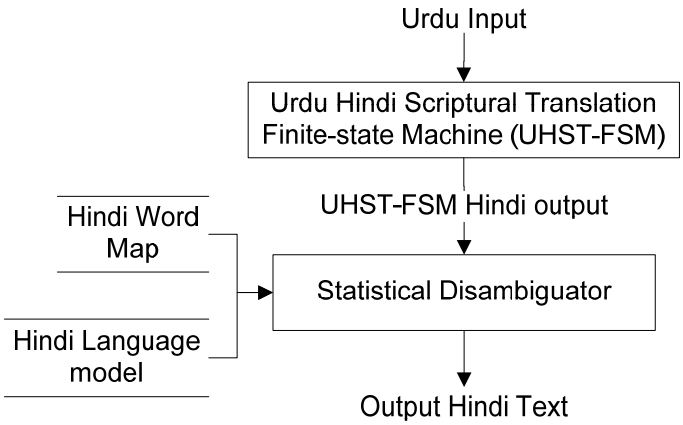


Figure 40. Hybrid model for scriptural translation

Now we discuss all components of our hybrid model for scriptural translation one by one. We will also show the effect of each step on some example Urdu sentences from HU Test Set 2 shown in Figure 41 with their Hindi reference. These examples have two parts: one contains necessary diacritics, and the other does not contain any diacritical mark.

Examples	(i) مَیں نے بہُت ادھِک کام نہیں کِیا ہَے (ii) میں نے بہت ادھک کام نہیں کیا ہے	(1)
	(i) کیِنْدریہ سطْر پر بھی اور راجْیہ سطْر پر بھی (ii) کیندریہ سطر پر بھی اور راجیہ سطر پر بھی	(2)
References	मैं ने बहुत अधिक काम नहीं किया है (I have not done a great work) केन्द्रीय स्तर पर भी और राज्य स्तर पर भी (Both at the central level and at the state level)	

Figure 41. Example of Urdu sentences with their Hindi reference for our hybrid model

3.2.4.1. Finite-state scriptural translation

The finite-state model was already described and discussed in full details in Chapter 2. So we will not discuss it again here. Figure 42 shows the Hindi translation of our example Urdu sentences using our finite-state system for Hindi-Urdu scriptural translation. We have highlighted the words that are not correct according our Hindi references.

(1) (i) मैं ने बहुत अधिक काम नहीं किया है
(ii) में ने बहत अधक काम नहें कया हे

(2) (i) केन्द्रीया स्तर पर भी और राज्या स्तर पर भी
(ii) केंदरया सतर पर भी ओर राजया सतर पर भी

Figure 42. UHT-FSM output

3.2.4.1.1. Statistical disambiguator

This component statistically disambiguates the multiple possible solutions given by the *word map* using the *statistical Hindi language model*. The whole process of statistical disambiguation is described in Figure 43.

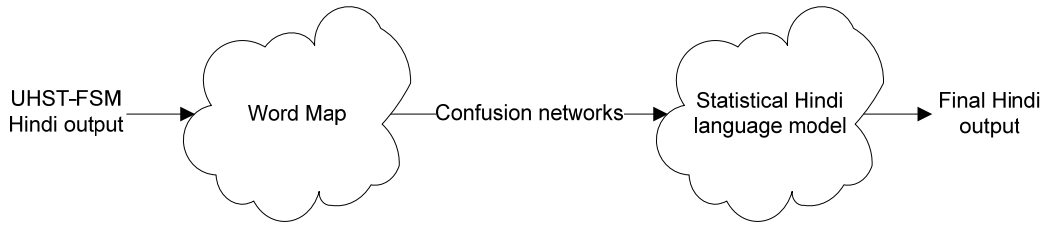


Figure 43. Statistical disambiguation process

We have used the 'disambig' utility of the SRILM toolkit [175] for producing the final Hindi output using the word map and statistical Hindi language model. The final output of our hybrid model is given in Figure 44 with the Hindi reference of each sentence. The wrong translation of a word is highlighted.

(1)	(i) मैं ने बहुत अधिक काम नहीं किया है
	(ii) मैं ने बहुत अधिक काम नहीं किया है
Hindi reference	(1) मैं ने बहुत अधिक काम नहीं किया है
(2)	(i) केन्द्रीय स्तर पर भी और राज्य स्तर पर भी
	(ii) केंद्रिय स्तर पर भी और राज्य स्तर पर भी
Hindi reference	(1) केन्द्रीय स्तर पर भी और राज्य स्तर पर भी

Figure 44. Example Hindi output of the hybrid model

In the UHST-FSM output, we had 12 wrongly translated words out of total 34 words. Our hybrid model for Urdu to Hindi scriptural translation wrongly translated only one word.

3.2.5. Experiments and results

For testing purposes, we have used HU Test Set 2 and HU Test 3, described above. Table 50 gives the results of our hybrid model on our test sets.

Test Set	Sentence Accuracy		Word accuracy	
	With diacritics	Without diacritics	With diacritics	Without diacritics
HU Test Set 2	14%	7%	85.8%	79.1%

Table 50. Results o hybrid model for HU Test Set 2

As compared to the results of our finite-state system for Urdu to Hindi scriptural translation, we have gained an increase of 26.1% and 6% in the word-level and the sentence-level accuracy for the Urdu input text without diacritics. We have also gained an increase of 1.9% and 4% in the word-level and the sentence-level for the Urdu input text with diacritics. Thus, with the help of target language resources, we can improve the quality of our systems for scriptural translation.

3.3. Comparisons of different models

In this section, we compare the quality of different types of computation models and systems for our scriptural translation. For comparison purposes, we devised two scales, one based on the word accuracy and the other based on the sentence accuracy that classify the quality of different systems for scriptural translation and consequently helps us to classify different systems. The scale was previously shown in Figure 27 and Figure 28 (page 43).

We will compare and classify different systems for scriptural translation according to these two scales.

3.3.1. Finite-state vs. SMT

We have developed 12 basic systems for each Hindi to Urdu and Urdu to Hindi scriptural translation. By tuning, we got another set of 12 systems. We could compare each and every system with the finite-state system for doing Hindi-Urdu scriptural translation. But here we have compared only the SMT systems with the best and the second best results with finite-state systems for each direction of Hindi-Urdu scriptural translation.

Figure 45 shows the comparison of the SMT systems with the finite-state system for Hindi to Urdu scriptural translation on all HU Test Sets.

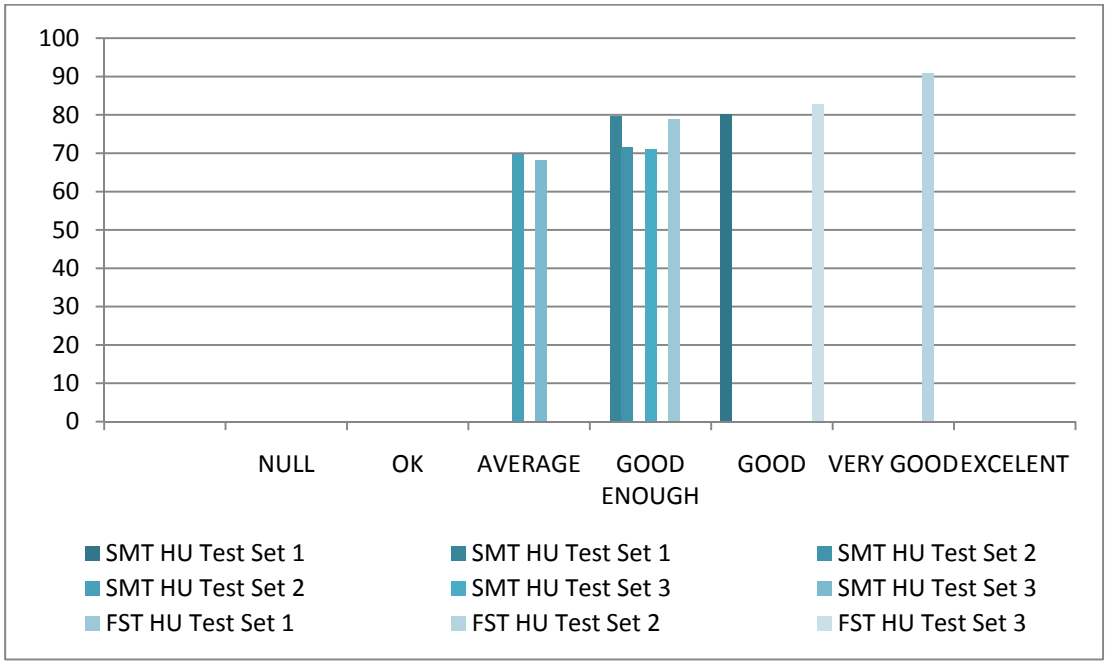


Figure 45. Comparison of SMT and FST systems for Hindi to Urdu translation

A similar comparison for Urdu to Hindi scriptural translation is shown in Figure 46.

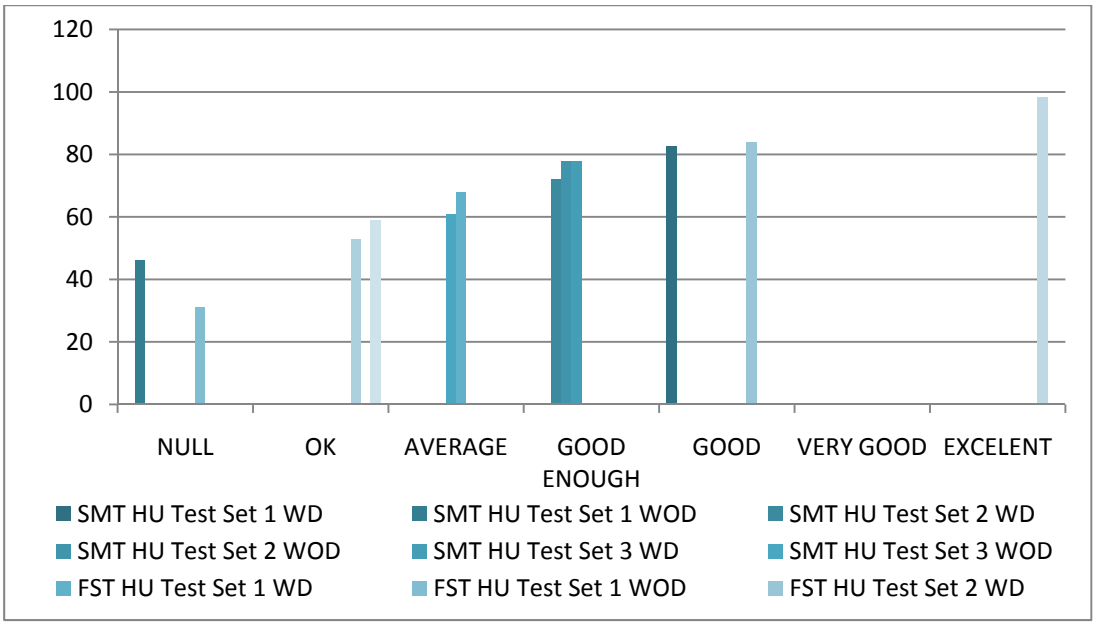


Figure 46. Comparison of SMT and FST systems for Urdu to Hindi translation

A comparison of Figure 45 and Figure 46 shows that the overall quality of Hindi to Urdu scriptural translation is better than the quality in the reverse direction. Although we have a system

that is categorized as excellent for Urdu to Hindi scriptural translation, we also have two systems that are categorized as NULL.

3.3.2. Finite-state vs. Hybrid

We have developed a hybrid model for Urdu to Hindi, so we have compared the quality of our hybrid model with the FST system for Urdu to Hindi scriptural translation.

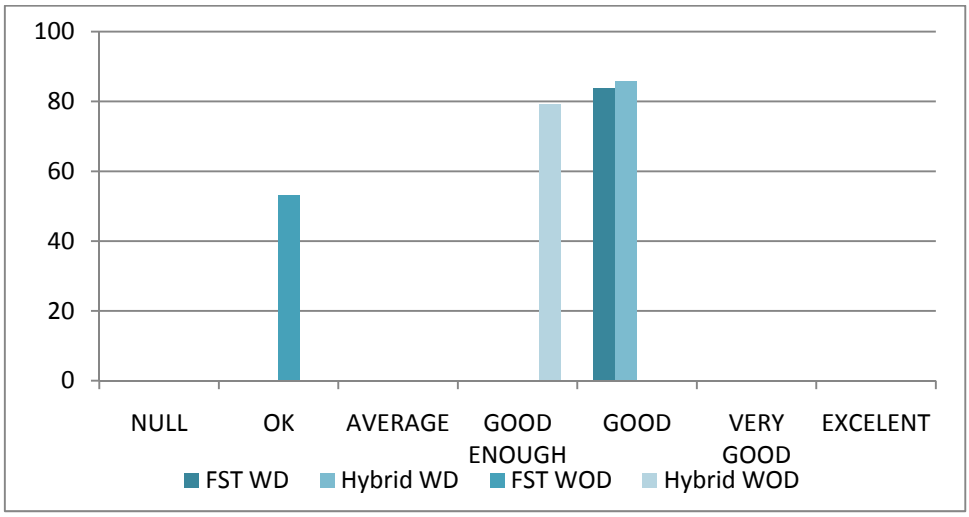


Figure 47. Comparison of FST and hybrid systems for Urdu to Hindi scriptural translation

We have gained an increase of 26.1% in case of Urdu input without diacritics with our hybrid system against the FST system.

3.3.3. SMT vs. Hybrid

We have compared the SMT systems and the hybrid system for Urdu to Hindi scriptural translation. Figure 48 shows the comparison graphically.

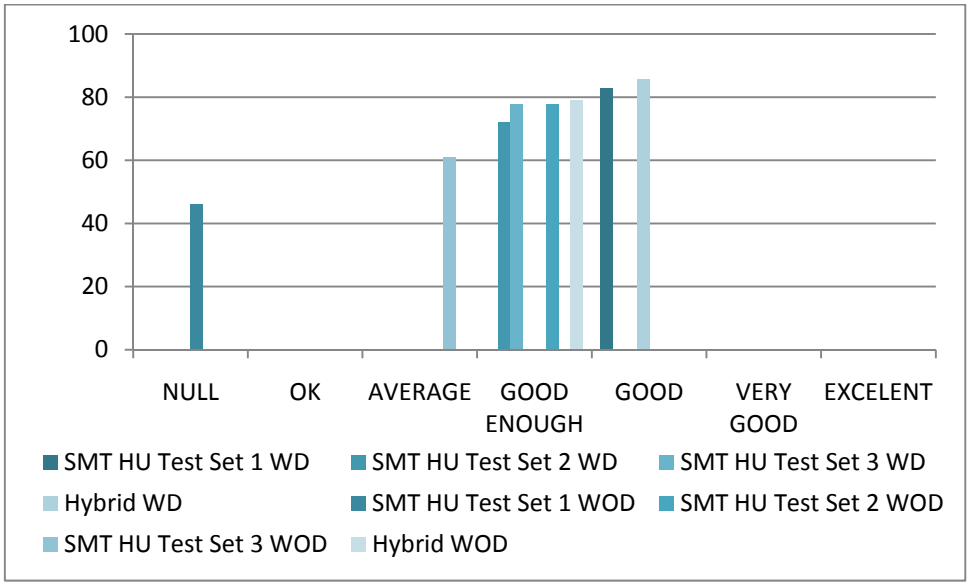


Figure 48. Comparison of SMT and hybrid systems for Urdu to Hindi scriptural translation

In both types of Urdu input with or without diacritics, the hybrid model outperforms the SMT systems.

3.4. Conclusion

Statistical and empirical methods are very successful in different fields of NLP. Our hypothesis was that they would help to produce more efficient and accurate systems. The results of scriptural SMT systems are comparable with the finite-state scriptural translation system at word-level, but they are much lower at sentence-level. In terms of user satisfaction, usability and effectiveness of the translation output, the scriptural SMT systems would also be classified very low by users in real scenarios.

The hybrid model outperforms the SMT systems in terms of sentence-level accuracy, user satisfaction, usability and effectiveness of the translation results. The hybrid model also works better than the finite-state scriptural translation systems.

Chapter 4. Interactive Scriptural Translation

The idea of interactivity is to use the successes of automated machine translation research and the human interaction together to make the translation results accurate and usable in real life. The MT results are not yet good enough to be used in real life, but MT research has seen a great number of successes during the past years. We can combine those successes of MT research with knowledge of MT users to produce translations with high precision and quality in an interactive way.

During machine translation research, we focus on the development (analysis, transfer and generation), but usually we do not consider the user perspectives like usability, acceptability, satisfaction and adequacy of the translation. The user is the target of our research, so we must give his perspectives a proper place.

First, we briefly discuss the state of the art in the field of *interactive machine translation*. Then we discuss the motivations behind the development of an *Interactive Scriptural Translation System* (ISTS). We also discuss the computational model of the interactive scriptural translation system. At the end, we describe our objective and subjective evaluations design, that is, how we will evaluate the usability, efficiency and effectiveness of our ISTS in real usage on internet.

4.1. Interactive Machine Translation

The concept of interactivity was first introduced in the late 1960s by Martin Kay and Ronald Kaplan in the MIND project (Management of Information through Natural Discourse) [18, 91]. They used the notion of interactivity during the analysis of the source text for performing MT when an expert user interacted with the system to disambiguate between ambiguous analyses. Following this concept, many researchers have developed interactive systems for various pairs of languages [17, 18, 22, 34, 123, 186, 192].

ITS (Interactive Translation System) is another important interactive system for MT [126-128]. It used man-machine interaction during the analysis and transfer phases of the automatic machine translation. The automatic translation output was then post-edited by a human translator. It was a multilingual and one-to-many translation system. It translated the source language English into the target languages Spanish, Portuguese, German, French and Chinese (a limited part).

ITS-2 [183] is based on the transfer-based architecture, with its three components of analysis, transfer and generation for French to English and vice versa. It introduced interaction at various levels of the translation process, like the lexicographic, syntactic, and lexical transfer levels.

[53-55, 109, 110] have brought an interesting focus shift to the field of Computer Aided Translation (CAT) in which interaction is directly aimed at the production of the target text, rather than at the disambiguation of the source text analysis. They introduced the *target text mediated* style for interactive machine translation.

Historically, CAT and MT have been considered different but close technologies [92]. In recent years, statistical approaches have also been used to develop IMT systems [11, 103].

4.2. Motivation

The idea of interactivity is not new to the field of machine translation. In words of [128],

> "To date, fully automatic translation has been shown to be commercially useful only when it is intended to be merely indicative (e.g. Russian-English MT at Rome Air Force Base) or when the system is tailored to a sub-language (e.g. TAUM-METEO). If the need is for high-quality translation of general text, the only possibilities seem to be (i) a highly successful large-scale AI approach, probably with a self-learning capability or (2) an interactive approach, with limited self-learning capability if possible."

These 30 years old observations seem to be true even in the present day. Following these observations, we suppose that we can also improve our scriptural translation systems with the notion of interactivity.

We have discussed and described finite-state, SMT and hybrid systems for scriptural translation of the Indo-Pak languages. In addition to the best translation results that are shown to the user, those systems also compute additional translation results that are not rated the best by the automatic system. We have gained a reasonably good, usable and satisfactory accuracy of approximately 20% *sentence accuracy* (80% *word accuracy*) and an average 20 minutes *post-editing* time per page with fully automatic finite-state scriptural translation. But we can improve these measures by cleverly utilizing the computed additional translation results with the help of our users in an interactive interface between the user and the translation system.

For example, we have mentioned that a native speaker of Seraiki applied our online Urdu to Hindi system[42] to translate a Seraiki/Shahmukhi book into a Seraiki/Devanagari book of more than 15,000 words. He spent 40 minutes per page for post-editing. We have already reduced this post-editing time to 20 minutes by developing a finite-state scriptural translation system for the Seraiki/Shahmukhi-Seraiki/Devanagari pair[43]. We expect that we can further reduce this post-editing time by using an intelligent interaction with the user. Therefore, we have designed an interactive scriptural translation system to interact with the user in the real scenarios and to take advantage of user feedbacks to improve the quality of the automatic scriptural translation systems. This improved automatic scriptural translation with added interactivity will thus reduce the user efforts to perfect the translated text.

Recent MT evaluation campaigns have been criticized because only tables of figures (such as BLEU, NIST, ORANGE, METEOR…) are shown as results, while these n-gram-based measures have been shown not to correlate well with human judgments [40]. Commercial MT systems have been consistently ranked low by these measures, while human judges ranked them quite high [81]. Thus, it is important to show real results by systems in operation, and to do subjective evaluations in addition to the objective evaluations.

4.3. Interactive Finite-state Scriptural Translation System

We have developed finite-state scriptural translation systems, discussed in Chapter 2, using Finite-State Transducers (FSTs). We have two varieties of FSTs. One is deterministic, and the other, non-deterministic, produces confusion networks for the input text. These non-deterministic finite-state transducers produce a set of possible solutions for each word of the input text. These sets of possible solutions contain the correct solution words more than 95% of the times. But the results of our finite-state system show that it correctly translates an input word approximately 80% (word accuracy) of the times. It is interesting to note that the chance

[42] http://www.puran.info/HUMT/index.html

[43] http://www.puran.info/saraikiMT/index.html

of calculating the correct word is more than 95%, but the accuracy of our system is only 80% at word level.

Generally, the accuracy of finite-state scriptural translation systems is approximately 10% to 20% at sentence-level (with roughly 80% word accuracy). For Hindi to Urdu scriptural translation of HU Test Set 2, the sentence accuracy is 27% (91% word accuracy). For Urdu to Hindi scriptural translation of HU Test Set 3 with diacritics, the sentence accuracy is 73.9% (98.4% word accuracy). Therefore, with the increase of word accuracy to more than 95%, we can increase the sentence accuracy from 10% to approximately more than 50%. It increases the usability, the acceptability and user confidence in our scriptural translation systems.

Instead of the user post-editing an incorrect word manually, it would take him less time to select the correct word (already computed) from the list of all candidates, provided that the correct word is among the first five choices, than to type it. In this way, we hope not only to improve the accuracy of our system from 10% to above 50% at sentence-level, but also to reduce the post-editing time of the user of our translation system. Consequently, we hope to gain more satisfaction and confidence of the user, and to improve the usability and effectiveness of the translations produced using our system. Therefore, we have developed an *Interactive Finite-state Scriptural Translation System* (IFSTS) by modifying the non-deterministic finite-state transducers discussed in Chapter 2.

4.3.1. System Architecture

The system architecture of the Interactive Finite-state Scriptural Translation System (IFSTS) is shown in Figure 49.

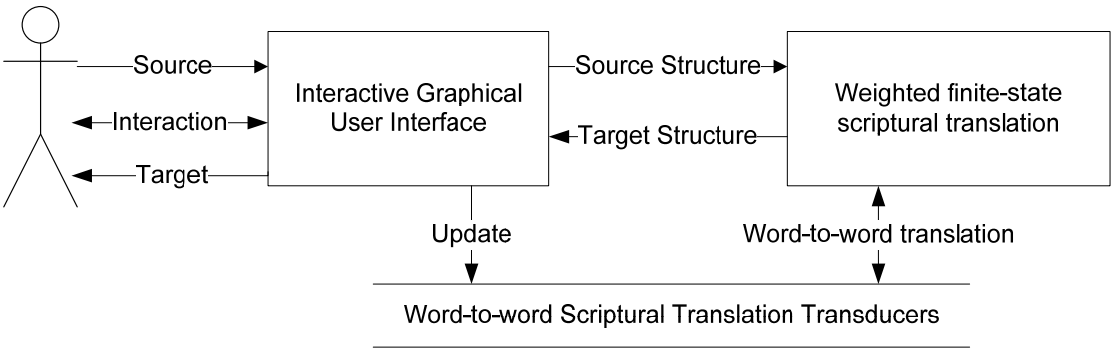


Figure 49. System architecture of interactive finite-state scriptural translation system

The interactive Graphical User Interface (GUI) is shown in Figure 50. The user can type or open a source text file for translation. Our system accepts a plain text file as an input file.

Figure 50. Interactive graphical user interface

The interactive GUI is a way to interact with the user. It takes the source text and the translation pair (source language/script and target language/script), given and selected by the user respectively. Then it builds a *source structure* that contains the source text and the translation pair and passes it to the underlying *weighted finite-state scriptural translation* component when the user presses the 'Translate' button. The weighted finite-state scriptural translation component translates the input text with the help of *weighted non-deterministic finite-state transducers* and *word-to-word scriptural translation transducers*. It builds a *target structure* that consists of *weighted confusion networks* for the target text and gives back the target structure to the interactive GUI.

The word-to-word scriptural translation transducers act as *Translation Memories* (TM), discussed later. The default output is taken to be the best path through the weighted confusion network and is shown to the user in the interactive GUI to correct it interactively. All ambiguous words are distinguished by highlighting them. The user interacts with the interactive GUI with the help of the mouse and the keyboard. S/he post-edits the translated text to her/his satisfaction. User interaction also helps to update the translation memories. We explain the user interaction later in Section 4.3.3.

In the remainder of this section, we discuss these components one by one.

4.3.2. Weighted finite-state scriptural translation

For the weighted finite-state scriptural translation, we have modified the non-deterministic finite-state transducers presented in Chapter 2 and introduced the notion of weights. We assign a weight of 1 to all unambiguous translation rules. For ambiguous translation rules, weights depend on the frequency analysis of corpora, explained later. Figure 51 shows the weighted finite-state scriptural translation and the word-to-word scriptural translation transducers components.

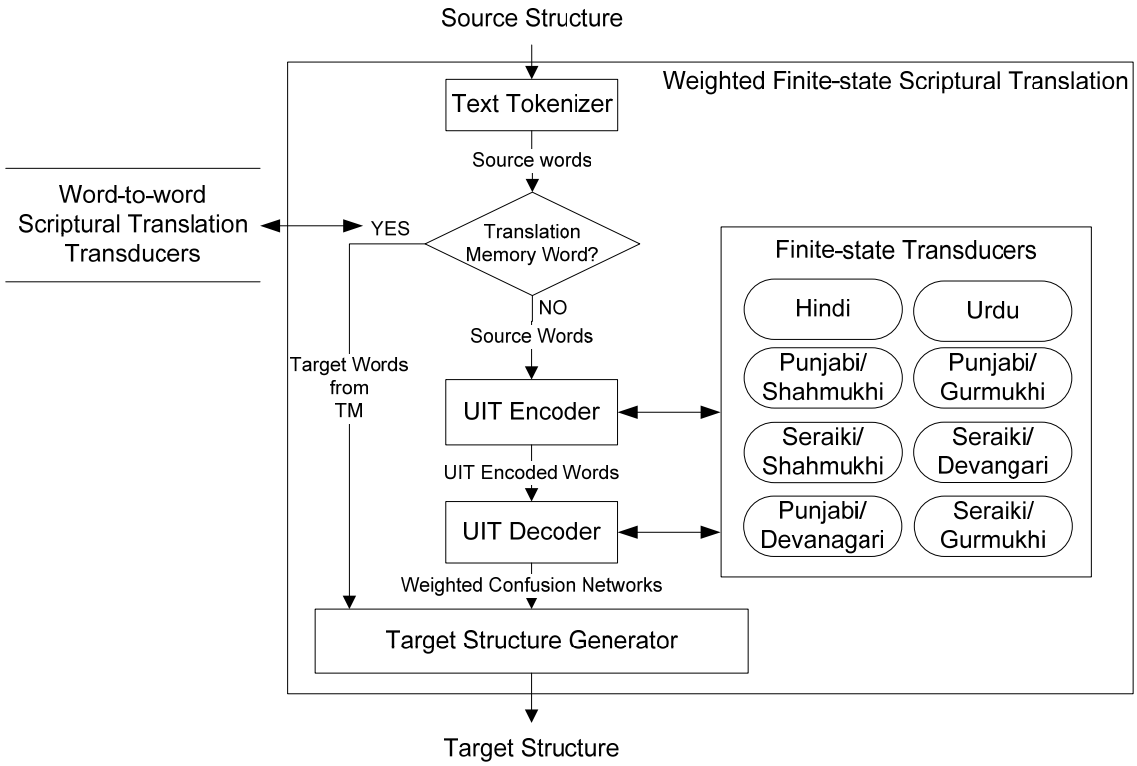


Figure 51. Weighted finite-state scriptural translation with the help of translation memories

The weighted finite-state translation component receives the source structure, built by the interactive GUI. The translation pair tells the source and the target language of the scriptural translation.

`Text Tokenizer` takes the source text from the source structure and segments it into its constituent words or source words. First of all, the source words are looked up into translation memories stored as finite-state word-to-word scriptural translation transducers. If a word exists in the translation memory, then it is directly converted into the equivalent target word using a word-to-word scriptural translation transducer.

If it does not exist in the translation memory, then it is converted into UIT with the help of a finite-state transducer (source language/script to UIT transducer) by the `UIT Encoder`. Then, each UIT-encoded word is converted by the `UIT Decoder` into a set of weighted possible solutions, as shown in the table below.

We represent these sets of possible weighted solutions in the form of weighted confusion networks. In case of ambiguities, the confusion network contains multiple weighted paths for each ambiguous word. Otherwise, it contains a unique path with a weight of 1.

We demonstrate the working of the IFSTS on the example sentence of Figure 6. Table 51 shows possible solutions for the example sentence with weights attached to each possible solution word.

दुनिया को अमन की ज़रूरत है [dʊnɪjɑ ko əmən ki zərurət̪ hæ] (The world needs peace)		
Text Tokenizer	**UIT Encoder**	**UIT Decoder**
दुनिया	dUnIjA1	[دُنیا (0.90) , دُنیہ (0.10)]
को	ko	[کو (0.80) , قو (0.20)]
अमन	@mn	[امن (1)]
की	ki	[کی (0.80), قی (0.20)]
ज़रूरत	zrurt_d	[زرُرت (0.0006), ضرُورت (0.00036), ذرُورت (0.00006), ظرُورت (0.000052), ژرُورت (0.0000005), …]
है	h{	[ہے (0.70), ۓ (0.30)]

Table 51. Sample run of weighted scriptural translation

Finally, the `Target Structure Generator` builds a target structure using the outputs of the `UIT Decoder` and the *word-to-word scriptural translation transducer*. The target structure is given to the interactive GUI that stores the structure in memory and interacts with the user for possible post-editing.

4.3.2.1. Weighted finite-state transducers

In the case of Hindi-Urdu scriptural translation, the weights for scriptural translation rules come from the analysis of an Urdu corpus that contains 412,249 words in total. We have shown an example words frequency analysis in Table 21 of Chapter 2. We reproduce the table here for the convenience. It shows the Urdu corpus frequencies of words that contain the characters shown in Table 52. It also gives *word frequency ratio*.

Hindi	**Urdu (corpus words)**	**Word frequency ratio**
त	ت (41,751), ط (1312)	ت (10.13%), ط (0.32%)
स	س (53,289), ص (751), ث (86)	س (12.93%), ص (0.18%), ث (0.02%)
ह	ہ (72,850), ح (1800)	ہ (17.67%), ح (0.44%)
ज़	ز (2551), ض (1489), ذ (228), ظ (215), ژ (2)	ز (0.62%), ض (0.36%), ذ (0.06%), ظ (0.522%), ژ (0.000485%)

Table 52: Frequency analysis of Urdu corpus

To assign a weight to each ambiguous translation rule between different states of our finite-state transducer, we have used the frequency analysis. For example, we have assigned, 0.0062 to the rule 'z → ز', 0.0036 to the rule 'z → ض' and so on. Similarly, we have assigned weights to all ambiguous translation rules in our finite-state transducers that convert a UIT encoded text into a target language text. A sample weighted finite-state transducer, developed from rules for the

characters shown in the table above from UIT to Urdu scriptural translation, is shown in Figure 52.

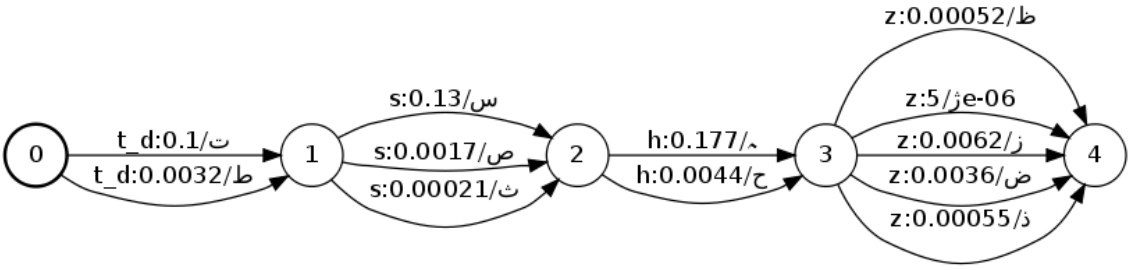


Figure 52. A sample finite-state transducer for weighted finite-state scriptural transaltion

These weighted finite-state transducers produce a weighted confusion network for each input word, and consequently for the input text that we use to interact with the user in our interactive GUI. A sample weighted confusion network for the example sentence (in Urdu, the output) is shown in Figure 53.

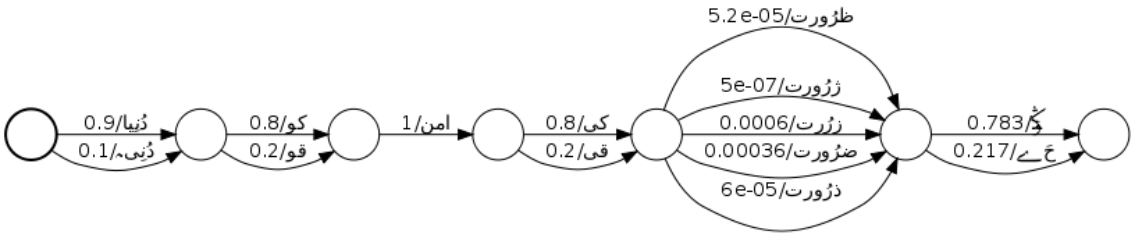


Figure 53. A sample weighted confusion network for the example sentence

4.3.2.2. Word-to-word scriptural translation transducers

In the case of Hindi-Urdu scriptural translation, we used a parallel word list of more than 55,000 words for developing the SMT systems for Hindi-Urdu scriptural translation, discussed in the previous chapter. Using the Hindi-Urdu parallel word list, we developed a word-to-word scriptural translation transducer for Hindi-Urdu scriptural translation. Table 53 shows a sample XFST code of that last transducer.

```
read regex [[अबिरोधी} -> [ابرودھی}, [अबरा} -> [ابرہ}, [आबरहन} ->
[آبرہن}, [अबरी} -> [ابری}, [अबरेषम} -> [ابریشم}, [अबरेषमी} ->
[ابریشمی}, इबरीक़ -> [ابریق}];
```

Table 53. Sample XFST code for word-to-word scriptural translation transducer

Figure 54 shows the sample word-to-word scriptural translation transducer for the XFST code, shown in the table above.

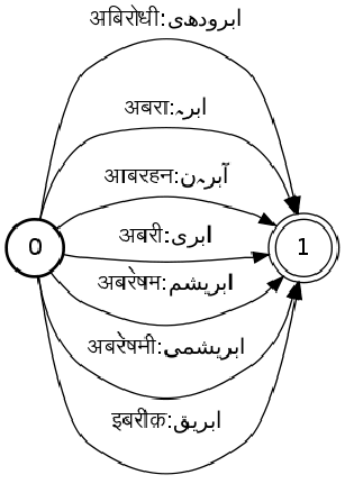


Figure 54. A sample word-to-word transducer

A word-to-word scriptural translation transducer is deterministic most of the times, but in some cases it is also non-deterministic and outputs a confusion network. A part of the Urdu to Hindi word-to-word scriptural translation transducer is shown in Figure 55. It shows the ambiguous nature of these word-to-word scriptural translation transducers.

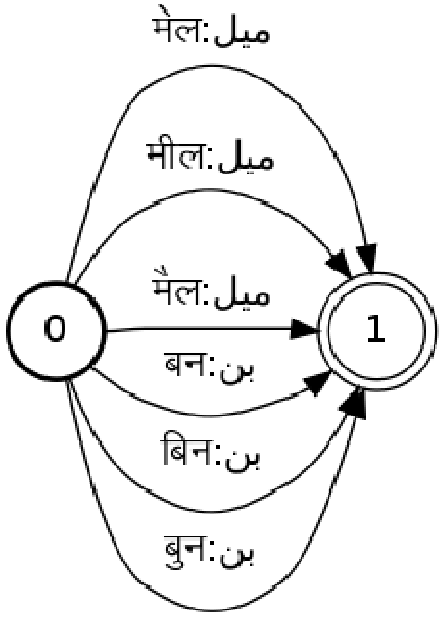


Figure 55. Ambiguous nature of word-to-word scriptural translation transducers

In the transducer above, the Urdu word میل can be translated into मील [mil] (mile), मैल [mæl] (dirt) or मेल [mel] (mail) in Hindi so that we have three possible solutions for this Urdu source word. In case of word-to-word translation, the maximum degree of ambiguity we have encountered during our work is 3. We could introduce weights to the confusion networks, produced by the word-to-word scriptural translation transducers, with the help of word frequency analysis of corpora. But we have not done this because the user has to select among the three possible solutions, which is not a big search space, in the interactive GUI.

In case of Hindi-Urdu scriptural translation, we start with 55,000 parallel words to build our word-to-word scriptural translation transducers. For other pairs of the Indo-Pak languages we do not have such parallel word lists. Thus we have developed a sample word list of 20 words for each pair and have developed word-to-word scriptural translation transducers for these pairs. These pairs are at the moment: Punjabi/Shahmukhi-Punjabi/Gurmukhi and Seraiki/Shahmukhi-Seraiki/Devanagari.

Because of interactivity, IFSTS will automatically enrich these word-to-word transducers for other language/script pairs when the interactive translation service will be used over the Internet. It will store all unseen source words and their post-edited translations into a text file, and then automatically generate the XFST code from these stored text files and compile them into word-to-word scriptural translation transducers. Finally, it will compose these newly built word-to-word transducers with the previous version of the word-to-word transducer and update the system with the new version of the word-to-word transducer for each language/script pair. Each time word-to-word transducers are updated, the accuracy of IFSTS will increase.

Word-to-word scriptural translation components for IFSTS are cost-effective in terms of computing time and space, as are finite-state transducers, used in the domain of NLP. The other main reason for developing finite-state word-to-word transducers is that our weighted scriptural translation component is also a finite-state system. Thus these two systems can communicate easily and are compatible with each other.

4.3.3. Interactive GUI

Figure 50 shows the interactive GUI of our IFSTS. We can open a text file or paste the source text in the text box to input text. After selecting the source and the target languages, we press the translate button to translate the source text.

For example, after typing the example sentence of Table 51 in the input text box and selecting Hindi and Urdu as the source and the target languages respectively, we have pressed the translate button. The interface changes as shown in Figure 56. By default, it shows the translation result generated from the best path through the weighted confusion networks of the target structure, in the output text area. The interactive GUI also highlights the ambiguous words to show them to the user during the post-editing task. The user interacts with the interactive GUI with the help of mouse and keyboard actions.

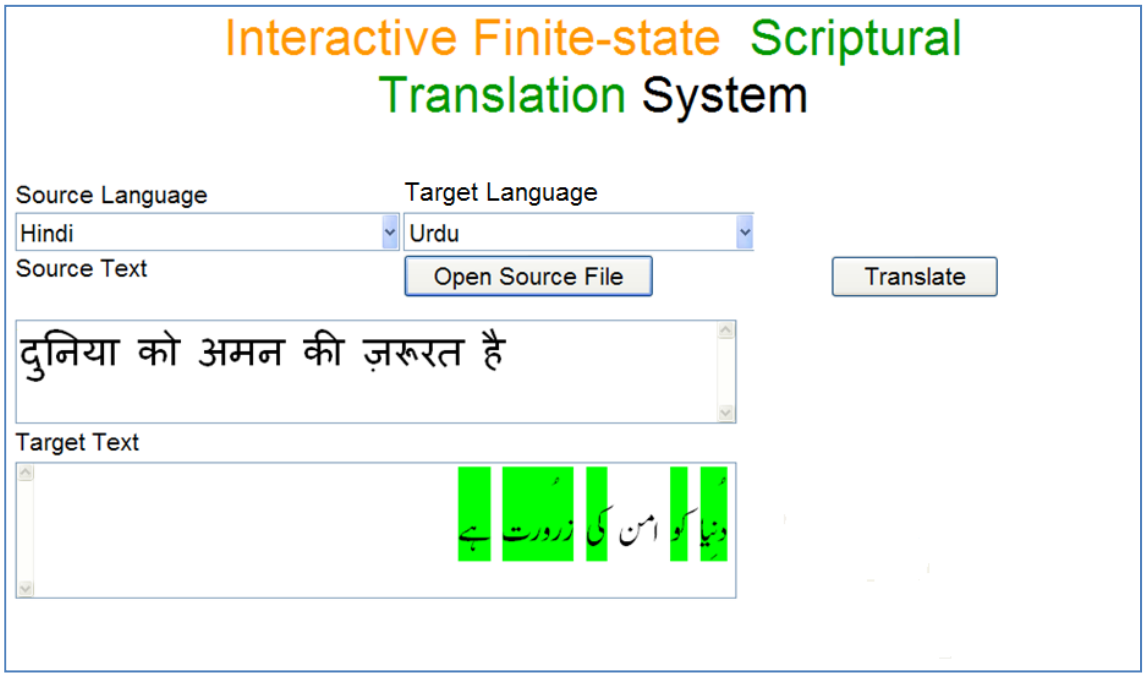


Figure 56. Interactive graphical user interface of IFSTS after translation process

When the user clicks on an ambiguous highlighted word, the interactive GUI shows to the user all possible ambiguities, which IFSTS has already computed during the automatic translation task, for the word in focus. The interface changes as shown in Figure 57.

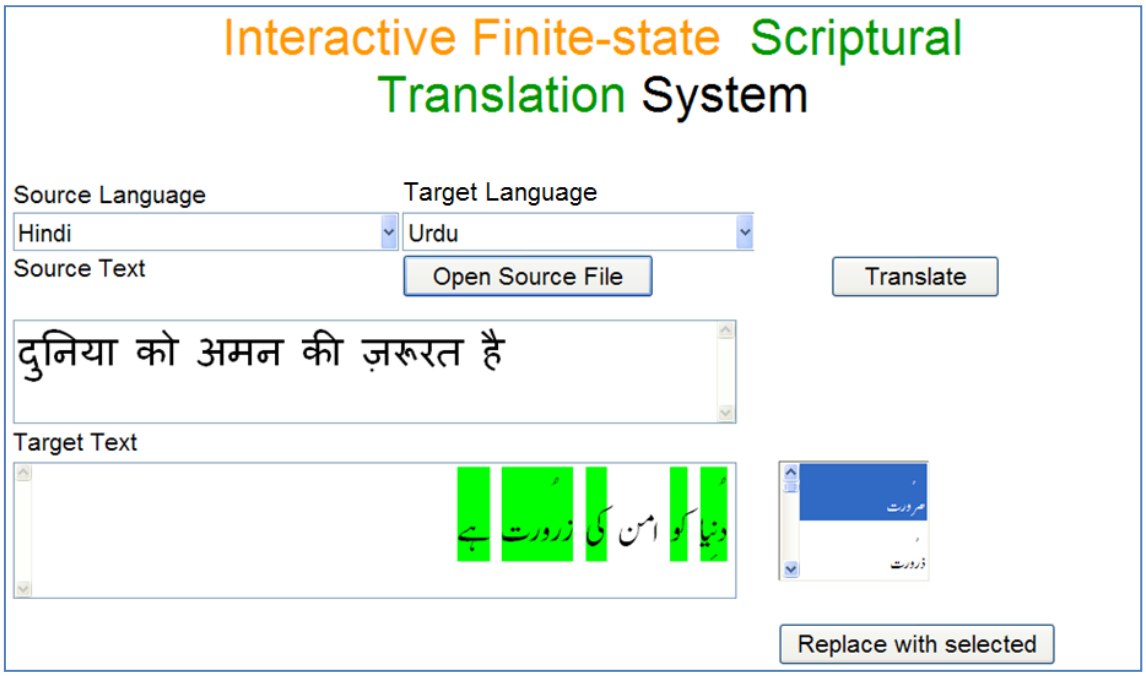


Figure 57. All possible word options in IFSTM

The user can replace the specific word in focus with a word from the list shown, or type the correct word directly.

It is desired that the correct solution, if computed, appears among the top 3 or 5 words such that the user does not have to look at a large number of words to find the correct solution. We have devised a learning approach that consists in automatically storing the unseen words and as a result compiling and updating the word-to-word scriptural translation transducers from time to time. In this way, we can accommodate the user feedback during interactive post-editing.

Once the system has added an unseen word to the word-to-word translation component, the user will see no ambiguity at all, or have to select the correct solution among a maximum of 3 words. But during the early system usage, the user has to seriously post-edit the text to increase the productivity of the interactive finite-state scriptural translation system and to reduce his post-editing time.

In the figure above, the correct word for the focused word in the output text is ranked first among all possible solutions. In average, for a good computer user, it takes approximately 10 to 15 seconds to type a word. This time may vary from user to user, but in any case, the time taken to replace the incorrect word from the output text with the correct word with three clicks is less than 5 seconds.

We automatically store the source and the post-edited translation results in order to develop parallel corpora for the Indo-Pak languages as a by-product. We also store the unseen words and their post-edited translations for developing parallel word lists for the language/script pairs of the Indo-Pak languages. These resources will be utilized in future for developing statistical translation systems for the said languages.

We also ask users to rank the quality of the translation of the IFSTS, once they have finished the post-editing task.

That interactive system is not available online yet, but we are planning to put it online in the near future.

4.4. Evaluation Design for Interactive Scriptural Translation Model

We have not tested or evaluated the interactive scriptural translation model in the real scenario, as we do not yet have Hindi, Urdu, Punjabi and Seraiki users to test the system and give us their feedback. But we will have them as soon as our system will be usable online. We have planned to undertake a comprehensive evaluation campaign with the help of colleagues in India and Pakistan in the near future by putting IFSTS on Internet.

For the evaluation design, we have considered different parameters, namely the *mouse* and *keyboard actions* to compute the efforts of users to post-edit the translated text to their satisfaction in the interactive system. For that, we record the total number of clicks and key strokes that each user has performed for post-editing. Another important measure is the post-editing time, that is, the time a user has spent to correct the result of automatic translation. The other objective evaluation measures which we have considered for the evaluation design of IFSTS, are character-level, word-level, and sentence-level accuracies. We have also considered edit distances between the automatic translation output and the post-edited translation.

In terms of subjective evaluation of IFSTS, we have considered user satisfaction, fluency, and adequacy of the translation. Table 54 shows different objective and subjective measures that we will use for the online permanent evaluation of our IFSTS for the Indo-Pak languages as an online Web service.

Objective measures	Subjective measures
Character Accuracies at characters, word and sentence level	User satisfaction
Edit distance at word and sentence level	Translation Fluency
Post-editing time	Translation Adequacy
Number of clicks by the user to do post-editing	Translation Usability
BLEU and NIST scores	

Table 54. Objective and subjective measures for IFSTS

IFSTS automatically computes the objective evaluation measures. On the other hand, subjective measures cannot be calculated automatically. For these measures, we will ask a series of questions to the user after post-editing of a translation. For example, we will ask him/her to associate a satisfaction measure with the translation that he has finished editing by clicking on the value corresponding to his opinion. The scale of user satisfaction is shown in Figure 58.

Not satisfied	*Slightly satisfied*	*Satisfied*	*Very satisfied*	*Totally satisfied*
0	1	2	3	4

Figure 58. Scale of user satisfaction

Similarly, we will use scales for other subjective evaluation measures, shown in Figure 59.

Usability				
Not usable	*Slightly usable*	*Usable*	*Highly usable*	*Completely usable*
0	1	2	3	4
Fluency				
Not fluent	*Slightly fluent*	*Fluent*	*Highly fluent*	*Completely fluent*
0	1	2	3	4
Adequacy				
Not adequate	*Slightly adequate*	*Adequate*	*Highly adequate*	*Completely adequate*
0	1	2	3	4

Figure 59. Scale of user satisfaction

After each post-edition in IFSTS, the user will be asked questions about the subjective measures, will select his answer to each question with simple clicks, and will submit it to the system. In this way, we will get real subjective measures for the scriptural translation systems for the Indo-Pak languages.

Part III
Interdialectal Translation–a Weak Translation Problem

Chapter 5. Interdialectal Translation – another Weak Translation Problem

MT between different languages, like Arabic, Chinese, English, French and Japanese, has been an attractive but extremely difficult and multifaceted research area since the 1950s. Translation is not only important between different languages but also between dialects of the same language. Interdialectal translation is another weak translation problem.

In this chapter, we discuss different problems of interdialectal translation. Then we perform an analysis of interdialectal translation and layout a computational model for it. We also discuss different tools and SLLPs for developing an interdialectal translation system. Finally, we discuss the scope of interdialectal translation.

5.1. Problems of Interdialectal Translation

Interdialectal translation is a *weak translation problem*, like scriptural translation, because there exists only one correct solution or a very limited number of correct solutions in the target dialect for a given sentence in the source dialect. For example, the French term '*congère*' must be translated into the Quebecois term '*banc de neige*' and vice versa. It is clear from the French–Quebecois example that we have to develop parallel lexicons for performing interdialectal translation between dialects of languages like French, English, Hindi, Punjabi and Urdu. Table 55 shows some specific subproblems of the generic problem of interdialectal translation with a few example instances and constraints.

Generic Subproblem	Specific Subproblems	Instances	Constraints
Interdialectal translation	Word-to-word translation Scripturasl translation Intralingual translation	Québécois–French	$SL = TL$ $SW = TW$
		English (USA)–English (UK)	
		Malay/Latin–Indonesian/Latin	
		Sindhi/Sindhi– Sindhi/Devanagari	$SL = TL$ $SW \neq TW$
		Punjabi/Gurmukhi– Punjabi/Shahmukhi	
		Malay/Jawi–Indonesian/Latin	
		Hindi/Devanagari–Urdu/Urdu	

Table 55. Subproblems of interdialectal translation

Interdialectal translation is computationally more complex and difficult than *scriptural translation* because it requires additional resources and linguistic knowledge. In the following two sections, we show that interdialectal translation is more complex and difficult than *word-to-word translation* using only a parallel lexicon. We also discuss the problem of under-resourcedness in the context of interdialectal translation.

5.1.1. Lexical divergence and word-to-word translation

We have partially discussed lexical divergence between dialects of a language in Chapter 1 during our analysis of scriptural translation. For a given word in the source dialect, we have to decide whether the word will be translated according to a bidialectal dictionary (word-to-word

translation) or whether it will be scripturally translated into the target dialect using scriptural translation [50, 97].

For French and Quebecois interdialectal translation, we have to develop an interdialectal lexicon for performing basic interdialectal translation. Unlike English, French differentiates between nouns, adjectives, adverbs and verbs on the basis of gender and sometime number. For example, the English noun phrase 'a car' is translated into 'une voiture' and the noun phrase 'a bicycle' is translated into 'un vélo', because the French nouns 'voiture' for 'car' and 'vélo' for 'bicycle' are feminine and masculine, respectively. Therefore, we have to translate the article 'a' into 'une' and 'un' respectively for these two French nouns.

In English, we do not differentiate between nouns, adjectives, adverbs and verbs on the basis of gender. Therefore interdialectal translation between dialects of English (English UK–English USA) is less complex than interdialectal translation of French (French-Quebecois). Table 56 shows a sample list of French–Quebecois parallel words.

French	Gender	Quebecois	Gender	English
Congère	F	Banc de neige	M	Snowdrift
Airelle à feuilles étroites	F	Bleuet	M	Blueberry
Moustique	M	Maringouin	M	Mosquito
Séjour	M	Salon	M	living-room
Maintenant	Adv	Présentement	Adv	Now
Voiture	F	Char	M	Car
Copain	M	Chum	M	Boy friend
Copine	F	Blonde	F	Girl friend
Boutique	F	Dépanneur	M	Convenience store

Table 56. Sample French - Quebecois parallel word list

There is sometimes a non-parallelism in number, in the case of collective nouns. A mere word-to-word translation is not sufficient for the interdialectal translation of languages like French, Spanish and Italian, in which the agreements of gender, number and/or person are difficult aspects. It requires more linguistic knowledge than found in a simple parallel lexicon. For the French–Quebecois interdialectal translation, our parallel lexicon must contain the morphosyntactic properties (gender, number…) of the parallel words in the lexicon.

Additionally, depending on the dialect or language pair, we have to perform syntactic analysis at the level of elementary phrases or even of complete sentences to accurately perform interdialectal translation, discussed in detail later. Thus, interdialectal translation is more complex than simple word-to-word translation and scriptural translation.

5.1.2. Under-resourcedness

Under-resourcedness is another problem when building an interdialectal translation system. We can find simple parallel lexicons for language pairs like French and English, as the one shown in Table 56, but it is difficult to find annotated parallel lexicons even for dialects of resource-rich languages.

When we consider under-resourced and under-studied languages, we cannot find even a simple parallel lexicon. For example, there does not exist a single parallel lexicon for Hindi–Urdu (at the end of 2009).

5.2. Analysis of Interdialectal Translation

For analysis purposes, we can divide the process of interdialectal translation into various parts. We demonstrate the analysis of interdialectal translation with the help of examples from the French–Quebecois and Hindi–Urdu dialect pairs. We also demonstrate that Hindi–Urdu inter-

dialectal translation is more difficult and complex than French–Quebecois interdialectal translation.

5.2.1. Word-to-word translation decision

The first step in interdialectal translation is to decide whether a word in the sentence of a source dialect is subject to word-to-word translation or only to scriptural translation. For the French–Quebecois pair, scriptural translation between dialects is not required because both dialects are written in the same writing system. On the other hand, it is essential for the Hindi–Urdu pair.

For word-to-word translation, we must have or develop (in case of non-existence) a lexicon of parallel words to produce a word-to-word translation for the source dialect into the target dialect of a language. Two example Quebecois sentences are shown in Figure 60 with their word-to-word translation in French. Equivalent or translated words are highlighted in the source and the target sentences. Adjustments due to gender differences of translated words are shown in italics in target sentences.

Quebecois sentences	***French word-to-word translation***
(1a) Mon char est noir.	(2a) *Ma* voiture est *noire*.
(1b) C'est un dépanneur.	(2b) C'est *une* boutique.

Figure 60. Examples of Quebecois to French word-to-word translation with gender adjustments

For the Hindi–Urdu pair, Table 57 shows some example parallel words.

Hindi	Urdu	English
विस्तार [vɪsʈar]	[vʊsaʈ] وُسعت	Expansion
मनुष्य [mənuʃjɑ]	[ɪnsɑn] اِنسان	Human
इच्छा [ɪʧʧʰɑ]	[xəvɑhɪʃ] خواہِش	Desire
उत्तर [ʊʈʈər]	[ʤəvɑb] جواب	Answer
स्तर [sʈər]	[səʈh] سطح	Level
साधारण [sɑɖɑrəɳ]	[ɑm] عام	Common

Table 57. Sample parallel lexicon for Hindi - Urdu

Figure 61 shows an example sentence for Hindi–Urdu interdialectal translation. For that pair, we have to perform scriptural translation in addition to interdialectal translation.

Hindi sentence	***Urdu word-to-word translation***
घर का विस्तार किया गया है	گھر کا وُسعت کیا گیا ہے
[gʰər kɑ vɪsʈar kɪjɑ gəjɑ hæ]	[gʰər kɑ vʊsaʈ kɪjɑ gəjɑ hæ]
The house is expanded	

Figure 61. Example of Hindi–Urdu word-to-word translation without agreement adjustments

5.2.2. Agreement recomputation

Examples of Figure 62 show the strict word-to-word translation without adjustments for gender. We have highlighted the words (nouns) that are translated into equivalent nouns in French. The Quebecois nouns in the source dialect sentences are masculine, while their French equivalent nouns are feminine. Thus both French translations contain errors, unacceptable for a French reader. The errors in the French translations are also highlighted in Figure 62.

Quebecois sentences	***French word-to-word translation***
(1a) J'ai un char noir.	(2a) J'ai *un* voiture *noir*.
(1b) C'est un dépanneur.	(2b) C'est *un* boutique.

Figure 62. Examples of Quebecois to French word-to-word translation without gender adjustments

In these examples, we have to recompute the *article-noun* and *adjective-noun* agreements for correctly translating the Quebecois sentences into French. We cannot recompute such agreements without the linguistic knowledge of the source and the translated lexemes. Therefore, we must have a parallel lexicon with morphosyntactic knowledge of the source and the target lexemes. Additionally, we must perform a syntactic analysis at the level of sentences or at least elementary phrases (noun phrase, adjective phrase, adverb phrase…) to recompute agreements between words.

Similarly, we have highlighted the Hindi and Urdu words (noun) in Figure 63. In the Hindi–Urdu pair, the problem not only exists in the noun phrase, underlined in Figure 63, but also in the verb phrase (also highlighted in the figure below) that depends upon the subject noun phrase. Thus there exist two types of agreement problems, short distance agreements and long distance agreements that we must handle to produce a grammatically correct interdialectal translation. Some agreement problems in the Urdu word-to-word translation are highlighted in Figure 63.

Hindi sentence	***Urdu word-to-word translation***
घर का विस्तार किया गया है	گھر کا وسعت کیا گیا ہے
[gʰər kɑ vɪsʈɑr kɪjɑ gəjɑ hæ]	[gʰər kɑ vʊsɑʈ̪ kɪjɑ gəjɑ hæ]
The house is expanded	

Figure 63. Example of Hindi–Urdu word-to-word translation

In the Hindi–Urdu example, the short distance agreement problem of the underlined subject noun phrase can be corrected by converting the case marker کا [ka] into کو [ko].

5.2.2.1. Short distance agreements

The short distance agreement problem between words appears locally within elementary phrases like in the Quebecois–French and Hindi–Urdu examples above. We list a few short distance agreements within elementary phrases in Table 58.

Sr.	Short distance agreements
1	Article-noun agreement
2	Adjective-noun agreement
3	Article-adjective agreement
4	Case agreements

Table 58. Short distance agreements

5.2.2.2. Long distance agreements

The correspondences of gender and number across phrases in the same sentence are more complex to handle in languages like French, Italian, Spanish, Hindi, Urdu and Punjabi. They require a full syntactic analysis of the sentence. We show examples of agreements across different phrases in the same French sentences in Table 59. The words with correspondences are in bold.

French Examples	English
Les **livres** que j'avais **reçus.**	The books I had received.
Nous sommes **allés** au cinéma.	We went to the cinema.
Les **voitures** ont été **lavées**.	The cars were washed.

Table 59. Examples of long distance agreements

A Hindi–Urdu example has already been given in Figure 63. The highlighted parts, which are not underlined, have correlation with the highlighted parts that are underlined in the Hindi and Urdu sentences. It is a subject-verb agreement problem in which the verb phrase depends on the subject nouns phrase. Two more examples of subject-verb agreements are shown in Figure 64.

Hindi sentence	***Urdu translation***
मेरी कार तेज़ चलती है	میری کار تیز چلتی ہے
[meri kɑr t̪ez ʧəlt̪i hæ]	[meri kɑr t̪ez ʧəlt̪i hæ]
English: My car runs fast.	
मेरा घोड़ा तेज़ चाता है	میرا گھوڑا تیز چلتا ہے
[merɑ gʰoɽɑ t̪ez ʧəlt̪ɑ hæ]	[merɑ gʰoɽɑ t̪ez ʧəlt̪ɑ hæ]
English: My horse runs fast.	

Figure 64. Example of Hindi–Urdu word-to-word translation

5.2.3. Partial or complete syntax analysis

For solving the short distance agreement within an elementary phrase, it suffices to perform a partial syntactic analysis at the level of elementary phrases. On the other hand, we have to perform a full syntactic analysis of a sentence for solving long distance agreements. The important point to note here is that we need to perform an analysis (partial or full) only when it is required. It means that we do not need to analyze every sentence. In other words, we have to analyze a sentence on demand.

For example, we do not need to perform syntactic analysis when the morphosyntactic properties of the translated words in a sentence are the same in the source and target dialects. In this case, a simple word-to-word translation is sufficient to produce a grammatically correct interdialectal translation. On the other hand, when the morphosyntactic properties of the translated words are different in the source and target dialects, we must perform a partial or full syntactic analysis.

5.3. Computational Models for Interdialectal Translation

In this section, we discuss computational models that we can choose to develop different components for an interdialectal translation system.

5.3.1. Word-to-word translation

For a word-to-word translation component of an interdialectal translation, we have different choices. We can store the parallel lexicon of all word forms in a database like MySQL, then we can search for each word in the database and decide that we will translate the word in question or not. For a small parallel word list, this solution might work very well. But this is not an adequate solution for searching each word in the database for an interdialectal system with a large parallel word list, because it will take a lot of time to search each word in the database and the system will be very slow. Furthermore, word-to-word translation is not a single process in the system. There are other complex and time-consuming computational steps like analysis and generation. Thus, this choice is not the best choice.

A *binary search tree* could be a good choice. We can compile a word-to-word dictionary in the form of a binary tree and store it in memory. A word can be accessed in logarithmic time in a binary search tree. Another good choice could be a *hash-table*, in which a word can be searched in a constant time.

A finite-state transducer could also be a good choice for developing a word-to-word translation component. Finite-state transducers are time and space effective. It could take the source word and translate it into the target word. A string-to-string transducer can easily perform the word-to-word translation.

Additionally for languages with rich morphology, we need a morphological analyzer for computing the morphosyntactic properties of the source and the target words. On the basis of these properties, we will decide whether we need or do not need to perform the syntactic analysis. We can develop a morphological analyzer for dialects of the translation language, the language for which we are performing interdialectal translation, in a finite-state paradigm. In this way, we can combine the finite-state transducer for word-to-word translation and the morphological analyzer to perform word-to-word translation, as well as to decide whether or not to perform syntactic analysis.

We do not need to do scriptural translation for the Quebecois–French pair, but we need to perform it for the Hindi–Urdu pair. We have already implemented the scriptural translation system for Hindi–Urdu in finite-state technology. It will be easy to integrate the scriptural translation system and the interdialectal translation system, if both are developed with the same technology. Therefore, finite-state technology is a good choice for developing the word-to-word translation component and the morphological analyzer with the specifications that we have laid down.

5.3.2. Partial or complete syntactic analysis

For interdialectal translation, the above given examples illustrate that short-distance agreements can be resolved with a partial syntactic analysis of elementary phrases for interdialectal translation. On the other hand, long-distance agreements require a complete syntactic analysis of the full sentences.

The first question is whether a partial or complete parser exists for a dialect of a language or not. For example, there exists no readily available parser for the Quebecois dialect of French, which is a resource rich language. In the case of Indo-Pak languages, there exist no readily available parsers for Urdu and Hindi. For Punjabi, Sindhi, Seraiki and Kashmiri, there are no parsers at all, according to the best knowledge of the author.

Building language parsers is a time-consuming and difficult task. It requires language knowledge and expertise. For example, it takes approximately one year to build a rule-based large vocabulary coverage parser for a language, if we already have language knowledge and expertise. Thus building a complete parser is out of scope of our current work, although it is our future prospect to build full parsers for the Indo-Pak languages.

There exist tools to derive a parser from annotated corpora (tree banks), like the Charniak parser [44]. The simplest way to learn a context-free grammar from a parsed corpus (a tree bank), is to read the grammar off the parsed sentences. A parsed sentence is shown in Figure 65.

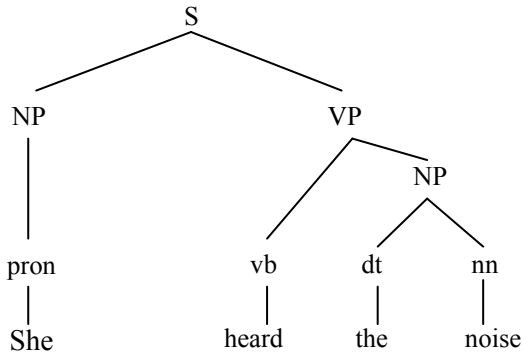


Figure 65: A parsed sentence [44]

From the figure above, we can learn the following context-free rules, shown in Table 60.

```
S → NP VP
NP → pron
Vp → vb NP
NP → dt nn
```

Table 60: A sample context-free grammarlearnt from tree bank [44]

In the case of the Indo-Pak languages, there exist no such annotated corpora (tree banks). Therefore, we cannot develop grammars in this way for these languages.

There remains the possibility to quickly build partial analysis structures in an expert way. This process requires several resources:

- a morphological analyzer.
- a segmenter.
- a monolingual dictionary containing morphological and syntactic features.
- an adequate SLLP to build a partial grammar that can be enhanced to build a complete grammar for syntactic analysis.

Finite-state technology is an adequate choice to build a morphological analyzer that can be integrated easily with our finite-state scriptural translation systems.

Building a segmenter for Indo-Pak languages is not very difficult task because their writing systems use spaces and punctuations for marking word and sentence boundaries, is not very difficult task. But it is a difficult and complex problem for language where word and sentence boundaries are not clearly marked, like Chinese, Japanese, Khmer and Thai. Building a monolingual dictionary is also a very time-consuming and difficult task and requires a lot of efforts in terms of money and human work.

For building a partial or full syntactic analyzer, there exist many grammar development tools like Xerox Linguistic Environment (XLE by Xerox)[44], Xerox Incremental Parser (XIP by Xerox Research Center Europe XRCE, Grenoble)[45], ROBRA [179] (Boitet, Ariane-G5 system), TELESI (Chauché, SYGMART 1985), *etc*.

XLE consists of algorithms for parsing and generating Lexical Functional Grammars (LFG) along with a rich graphical user interface for writing grammars. XIP is an on-the-fly rule compiler with a very powerful formalism, and an API to integrate its syntactic and semantic text parsing functionalities into end-user applications. It comes with an interface for grammar developers to build upon XRCE's XIP grammars. XIP includes finite-state technology and much more (arbitrary transformations). It seems to be a good choice for developing syntactic analyzers that can be integrated with our finite-state translation systems (scriptural and word-to-word) for the Indo-Pak languages.

5.3.3. Tree transduction and agreement recomputation

After the decision that we have to perform the syntactic analysis partially or fully, we create a partial or full syntax tree for the elementary phrase or the full sentence of the source dialect. Then we transform the source dialect syntax tree into the target dialect syntax tree with correct agreements between words in the target side. Finally, we generate the target dialect text from this transformed syntax tree.

Tree transducers were independently introduced by Rounds [163] and Thatcher [176] as a generalization of finite-state transducers. The Rounds/Thatcher tree transducer is very similar to a

[44] http://www2.parc.com/isl/groups/nltt/xle/

[45] http://www.xrce.xerox.com

left-to-right FST, except that it works top-down, pursuing sub-trees independently, with each sub-tree transformed depending only on its own passed-down state [62, 63].

A tree transducer T compactly represents a potentially infinite set of input/output tree pairs: exactly those pairs (T_1, T_2) for which some sequence of productions applied to T_1 (starting in the initial state) results in T_2. This is similar to an FST, which compactly represents a set of input/output string pairs; in fact, T is a generalization of an FST.

The tree transducers can be learnt from a finite training set of input and output tree pairs [62, 63]. But for automatic learning, we do need a training set that contains input and output tree pairs. There exists no such training data for French–Quebecois, English (UK)–English (USA), and Hindi–Urdu. Thus, we cannot learn these tree transducers automatically.

The other way to develop tree transducers is the expert way. We manually develop transduction rule grammars and then apply these grammars on an input source language syntactic tree to produce a target language syntactic tree. There exist different tools to implement these tree transduction grammars, discussed previously in section 5.3.2.

5.3.4. Syntactic translation model for interdialectal translation

Figure 66 shows the proposed interdialectal translation model.

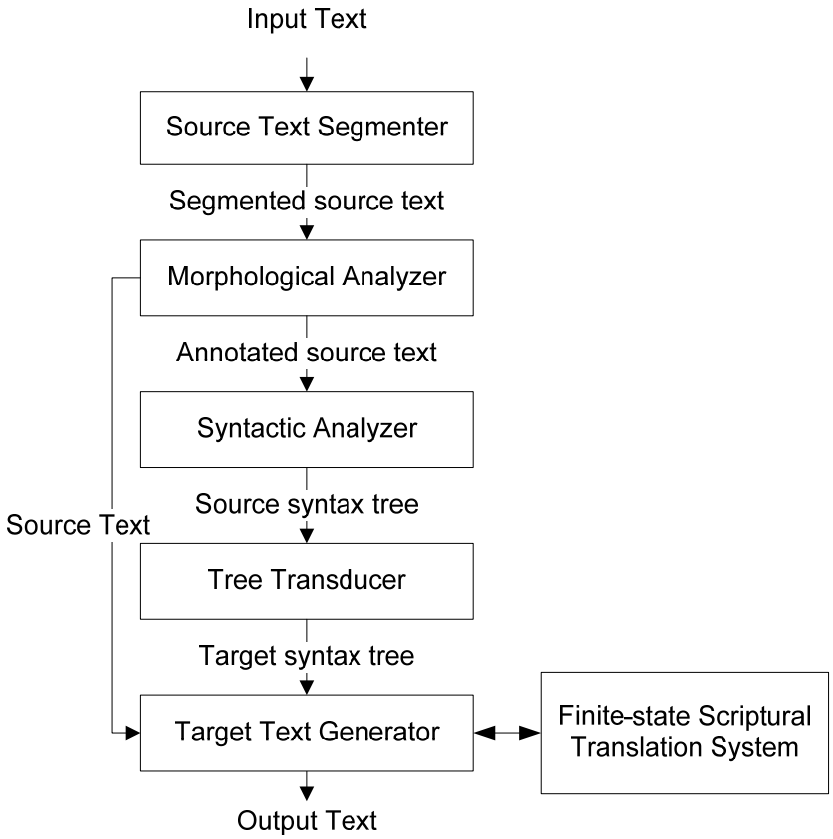


Figure 66: Proposed interdialectal translation model

In the proposed interdialectal translation model, an input text is received by `Source Text Segmenter`, which segments the source text and passes the segmented source text to `Morphological Analyzer`.

`Morphological Analyzer` is a crucial component of our interdialectal translation model. It not only decides that a word will be word-to-word translated or scripturally translated, but also decides that a syntactic analysis is required or not. If a syntactic analysis is not required, then it directly passes the segmented source text to `Target Text Generator` that gene-

rates the target text, using `Finite-state Scriptural Translation System`, if necessary. Otherwise, `Morphological Analyzer` passes the annotated source text to `Syntactic Analyzer`.

`Syntactic Analyzer` performs the partial or full syntactic analysis and generates the source syntax tree. This source syntax tree is then passed to `Tree Transducer`, that performs the source to target syntax tree transformation and generates the corresponding target syntax tree. Finally, `Target Text Generator` produces the output text using `Finite-state Scriptural Translation System`, if necessary.

5.4. Scope of Interdialectal Translation

Interdialectal translation is important for languages like English, French, Italian and Spanish. It is even important and inevitable when the two dialects in question use two unintelligible writing systems like Hindi–Urdu, Punjabi/Shahmukhi–Punjabi/Gurmukhi and Sindhi/Sindhi–Sindhi/Devanagari. The Hindi–Urdu pair alone represents a population of more than 1000 million people (including native and second languages speakers) around the word and they require an automatic way to read the books, journals, newspapers and websites published in the other script. Thus it is worthwhile to develop interdialectal systems for these languages.

Chapter 6. Towards an Interdialectal Translation System

In the previous chapter, we have given an analysis and discussed computational problems that are important for building an interdialectal translation system. We have laid down a model based on an expert computational approach, and on a linguistic architecture based on syntactic transfer. We have not yet implemented an interdialectal system in its totality, but we have developed some necessary parts that will be combined together to build an interdialectal translation system. Thus we are moving towards building an interdialectal translation system in the near future.

In this chapter, we present a preliminary study of two approaches for developing an Inter-Dialectal Translation System (IDTS). One is the *empirical approach* (SMT) using direct linguistic architecture, and the other is the *syntactic translation approach*, using an expert (rule-based) computational architecture. We also discuss the practical problems like data scarcity and building necessary resources (when they does not exist) that will be used to build the translation system in the future.

6.1. Approaches for Building an IDTS

In this section, we examine two approaches to build an IDTS.

6.1.1. Empirical approach

Empirical approaches are data driven approaches, requiring large amounts of data for building a translation system between a language or dialect pair. This approach is very fast and requires not prior linguistic and expert knowledge of the language pairs. In other words, it is a robust, fast and language-independent approach. In 1949, Warren Weaver introduced the idea to deal with the problem of MT with *statistical methods* and *information theory* [32, 182]. Brown *et al.* revived this idea in 1990 and laid down a statistical approach to MT [32]. An extract of their paper is given below:

> "The field of machine translation is almost as old as the modern digital computer. In 1949 Warren Weaver suggested that the problem be attacked with statistical methods and ideas from information theory, an area which he, Claude Shannon, and others were developing at the time (Weaver 1949). Although researchers quickly abandoned this approach, advancing numerous theoretical objections, we believe that the true obstacles lay in the relative impotence of the available computers and the dearth of machine-readable text from which to gather the statistics vital to such an attack."

In the present day, the field of statistical methods in MT has matured. A large number of data-driven and language-independent algorithms have been developed. SMT has been used to develop statistical (more exactly, probabilistic) translation system between various language pairs like English–French, English–Spanish, English–Arabic, French–Spanish, *etc*., for which a large amount of parallel data is available.

In the case of interdialectal translation, the power of modern PCs is sufficient to build SMT systems, but the scarcity of parallel data is a major and vital problem. Parallel data for dialects of a

resource-rich language like English and French either do not exist or exist in a very small quantity. In the case of under-resourced languages, there are no parallel data. For example, we have not found a single (even very small) machine-readable parallel lexicon or corpus for any dialect pair of Indo-Pak languages. Thus it is out of question to use empirical approach immediately for developing interdialectal translation systems in this context.

6.1.2. Syntactic translation approach

Following [21, 25, 26, 28, 178, 179, 190], we adopt and adapt the framework for *syntactic translation* to solve the problem of interdialectal translation in an expert manner. We have already shown the proposed syntactic translation model for interdialectal translation in conjunction with syntactic transfer in the previous chapter. We use *interlingua* and *transfer-based* linguistic architecture, shown in Figure 67.

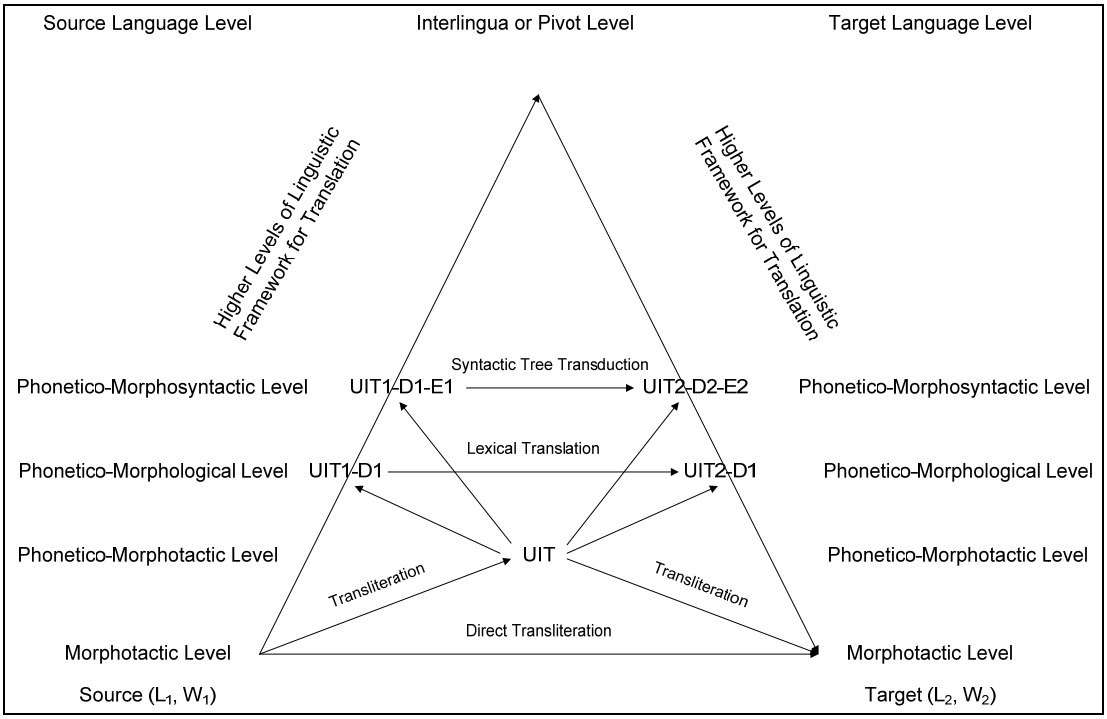


Figure 67: Adapted Vauquois' triangle

At the morphotactic and phonetico-morphotactic levels, we have already developed scriptural translation systems for Indo-Pak languages.

For higher levels, first we need to develop analysis systems on the source side and synthesis systems on the target side. At the phonetic, morphological and lexical translation levels, we have to develop interdialectal parallel word lists and morphological analyzers.

In the case of Indo-Pak languages, there exist a large number of studies on the morphology of Hindi [7, 8, 42, 158, 160, 170]. For Urdu, there also exist some studies on its morphology [19, 20, 37, 38, 75-77, 159]. In both cases, there is no readily available morphological analyzer that can be used freely for research purposes. On the other hand, there exists no study on the morphology of the other Indo-Pak languages.

At the phonetic and morphosyntactic levels, there is also some prior work on the syntactic analysis of Hindi and Urdu, but we found no such work for the other Indo-Pak languages. Therefore, in the absence of all these necessary components, we cannot check the performance and effectiveness of our proposed syntactic translation model for interdialectal translation. Also, it was not possible to develop all these linguistic and computational resources during our PhD.

In the following section, we discuss how we can use our previous systems for scriptural translation for Indo-Pak languages to develop useful linguistic and computational resources interactively on Internet.

6.2. Building Parallel Resources for Interdialectal Translation on Web

We have developed an interactive scriptural translation system, discussed in Chapter 5. This interactive system will be used in multifaceted ways. It will help to evaluate subjectively and objectively the performance of our scriptural translation systems. With the help of users of our online interactive scriptural translation system, it will also be used to develop different parallel resources that will be used to develop future statistical and syntactic interdialectal translation systems.

6.2.1. Parallel lexicon

During the interaction between our online scriptural translation system and its users and post-editing process, we also ask users to give an equivalent word for a translated word to produce a good localized translation in the desired target dialect. Figure 10 shows an example of Hindi–Urdu pair for interdialectal translation. For convenience, it is reproduced here.

भगवान	तुमहारी	**रक्षा**	करे
[**bʰəgəvɑn**	t̪ʊmhɑri	**rəkʃɑ**	kre]
کرے	حفاظت	تمہاری	اللہ
[kre	**hɪfɑzət̪**	t̪ʊmhɑri	**əllɑh**]

God may protect you.

The words in bold in the Hindi sentence are translated into the bold words in the Urdu sentence.

In the interactive interface, we can type the Hindi sentence in the source text field. After selecting Hindi as the source language and Urdu as the target language, we pressed the `Translate` button. The interface looks like the Figure 68.

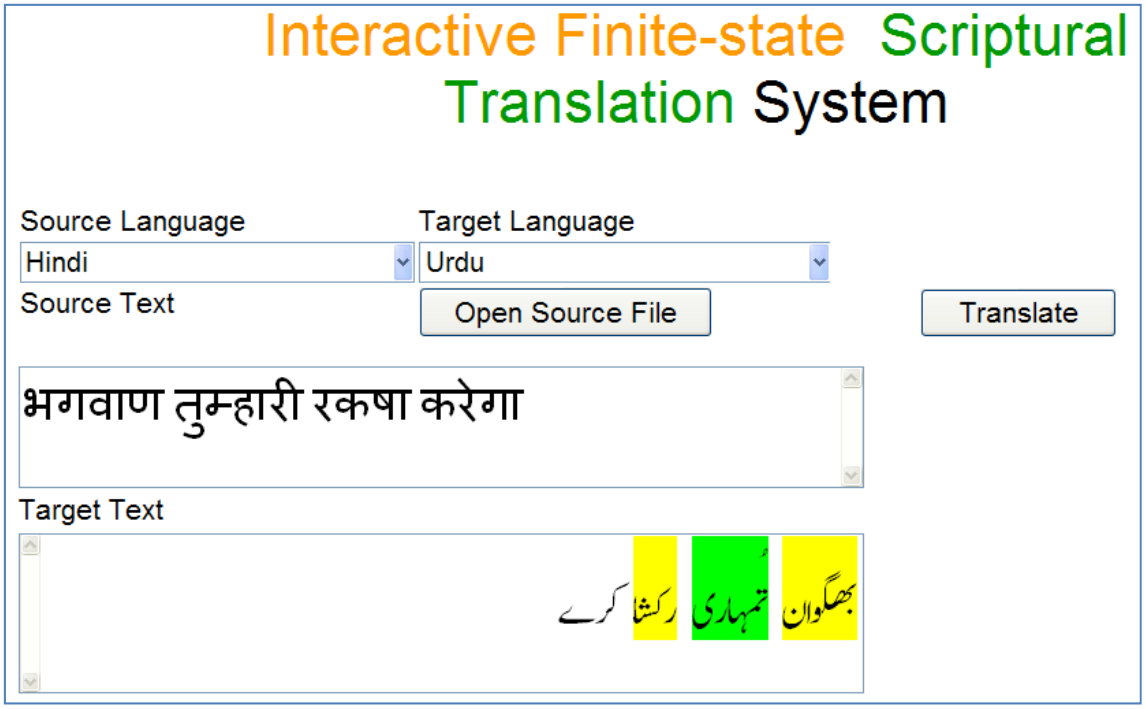


Figure 68: Interactive scriptural translation interface after automatic translation process

The first translated Urdu word is not ambiguous, but requires localization of the source word in the Urdu dialect. If the user wants to change this word, he can click on the word and the interface changes as shown in Figure 69. The interface provides a means to type in an equivalent word in the target dialect. This process serves two objectives. First, the user gets his desired output and the system gets a good satisfaction measure from him. Second, our Hindi–Urdu parallel lexicon grows, if it does not yet contain the source Hindi word and that is beneficial because it will be used to develop later a Hindi–Urdu *interdialectal* translation system.

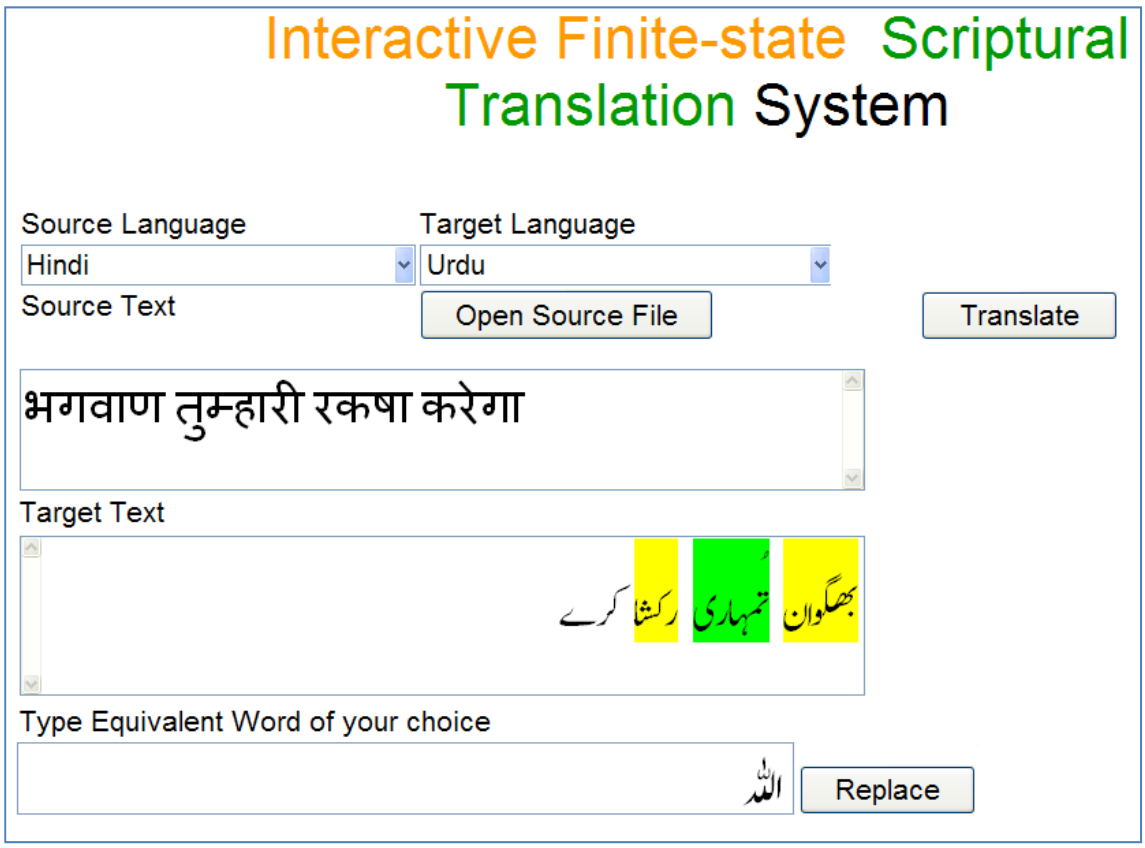


Figure 69: Interactive scriptural translation interface for adding equivalent words

This is a slow process for developing parallel lexicons for dialect pairs, but after a few years, we will have a parallel lexicon large enough to serve our purpose for the development of interdialectal translation systems.

The other important factor to keep in mind is that we may also find some non-serious users who can spoil the efforts of serious online users. To deal with this problem, we have developed two different instances of our interactive scriptural translation system. The first instance is freely available on Internet and everybody can use it. The other instance is dedicated to trusted and serious users and requires its users to login. A serious user on Internet can communicate with the system administrator to get her/his login by providing required information to the administrator.

6.2.2. Parallel corpus

The process of development of a parallel lexicon has been discussed above. During the same process, the interactive scriptural translation system also stores the source and the translated target text after post-editing by the user in a translation memory. This translation memory consists of parallel terms and parallel sentences. It not only helps to improve the translation quality of the currently available system, but will also serve the future development of statistical scriptural and interdialectal translation systems.

6.3. Building Linguistic Components

For the syntactic translation model for interdialectal translation, we require computational linguistic systems like morphological analyzer, syntactic analyzer and syntactic transformation component.

6.3.1. Morphological analyzer

There exist some prior studies on the morphology of the Hindi–Urdu pair [7, 8, 19, 20, 37, 38, 75-77, 158-160, 170], but a readily usable morphological system is not available yet. As discussed above, finite-state technology is suitable for the development of a morphological analyzer in our case. Like finite-state technology [89, 131-135, 161, 162, 166, 167], finite-state morphology is also very well documented and mature [12, 90, 161, 162].

In the presence of previous studies for the Hindi–Urdu pair, it is possible to build a finite-state morphological analyzer for each language of the Hindi–Urdu pair in three to four months with a limited vocabulary. But there does not exist any detailed study on morphology for the other pairs of Indo-Pak languages. Therefore, it may take a year or more to develop a morphological analyzer for an Indo-Pak language. Developing morphological analyzers for Indo-Pak languages is one of our future plans.

There exist various finite-state engines that can be used for the development of a morphological analyzer for a language in finite-state paradigm like ATEF (by J. Chauché, available at GETALP), OpenFST (by Google Research and New York University), FSM (by AT&T labs), Q-Systems (A. Colmerauer, available at GETALP) and XFST (by Xerox). ATEF is computationally very powerful for implementing a finite-state morphological analyzer [29, 30, 179], as it can handle not only flexional morphology, but also derivational and compositional morphology, and offers a very powerful way to define a full subgrammar to handle unknown words.

OpenFST and FSM are also very powerful finite-state engines, but it a bit hard to code a finite-state morphological analyzer in these systems, as they are general-purpose finite-state engines and do not contain specific features to facilitate the development of a morphological analyzer.

We chose XFST to implement morphological analyzers for Indo-Pak languages because we have used XFST for developing our finite-state scriptural translation systems and we are familiar with this finite-state engine. It is very well documented in [12] and also contains a good SLLP (specialized language for linguistic programming) for developing a finite-state morphological analyzer.

6.3.2. Syntactic and Syntax tree transformation grammars

Building parsers, syntactic analyzers and syntax tree transformation grammars is our long-term future commitment with Indo-Pak languages. Some studies about the syntax and parsing of Hindi–Urdu pair are available, but there exist no concrete parser for these languages. The other Indo-Pak languages also lack in this aspect of computational linguistics.

Following our choice of XFST for morphological analyzers, we can select Xerox Linguistic Environment (XLE) by PARC or Xerox Incremental Parser (XIP) linguistic engines for developing syntactic analyzers for Indo-Pak languages. These two systems can also be used to develop tree transformation or tree transduction grammars for the syntactic translation process at higher levels of the linguistic translation framework.

We selected XIP for implementing syntactic analyzers and tree-transduction grammars for Indo-Pak languages because we have a complete access to XIP software and its documentation due to the long-standing collaboration of our lab with Xerox Research Center Europe (XRCE)[46].

6.4. Conclusion

Interactive online scriptural translation systems will be used to overcome the problems that have restrained us from developing a practical interdialectal translation system. We have devised ways to build parallel lexicons and corpora to develop statistical interdialectal translation systems in the future. It will take some years to gather necessary and sufficient parallel resources.

We have also devised a theoretical syntactic translation model and have identified different linguistic components that are required to develop an interdialectal translation system. Although we have not yet developed a practical interdialectal translation system, we have developed theoretical translation model and a future road map for the development of many interdialectal translation systems for pairs of dialects or of very similar languages or sublanguages.

[46] http://www.xrce.xerox.com/

Conclusion and Perspectives

MT history is more than half a century old. The field of MT started to emerge with the advent of computer systems. In the beginning, it was used to translate Russian texts into English for defense and intelligence purposes. It was a very hot research area in the past and it is also a very central and highly funded area in the present. The need of MT is increasing day by day because the world is becoming a global village.

In general, a given sentence S of n words in the source language SL may have an exponential number of valid translations, say $N = k^n$ for some k. A *weak translation problem* is a translation problem for which N is very small, say less than 5 or almost always 1. In this study, we restricted our scope to the study and analysis of *weak translation problems* only.

We adopted a step-by-step approach to deal with weak translation problems. We started with the scriptural translation problem that seemed to be a computationally less hard and relatively simple problem. In the current literature, the terms transliteration and transcription are often confused. We have defined and proposed the new term 'scriptural translation' to denote a combined process of transliteration and/or transcription.

We were optimistic at the start that we would not only be able to solve the scriptural translation problem, but also the much harder and complex problems of interdialectal translation and of translation between closely related languages.

Actually, the study and analysis of the scriptural translation problem proved to be much more complex and hard than our preliminary estimates. We have experimented with finite-state, statistical and hybrid models to solve the scriptural translation problems. The graph of Figure 70 shows a brief comparison of results obtained on HU Test Set 2 using these three different models for the Urdu to Hindi scriptural translation.

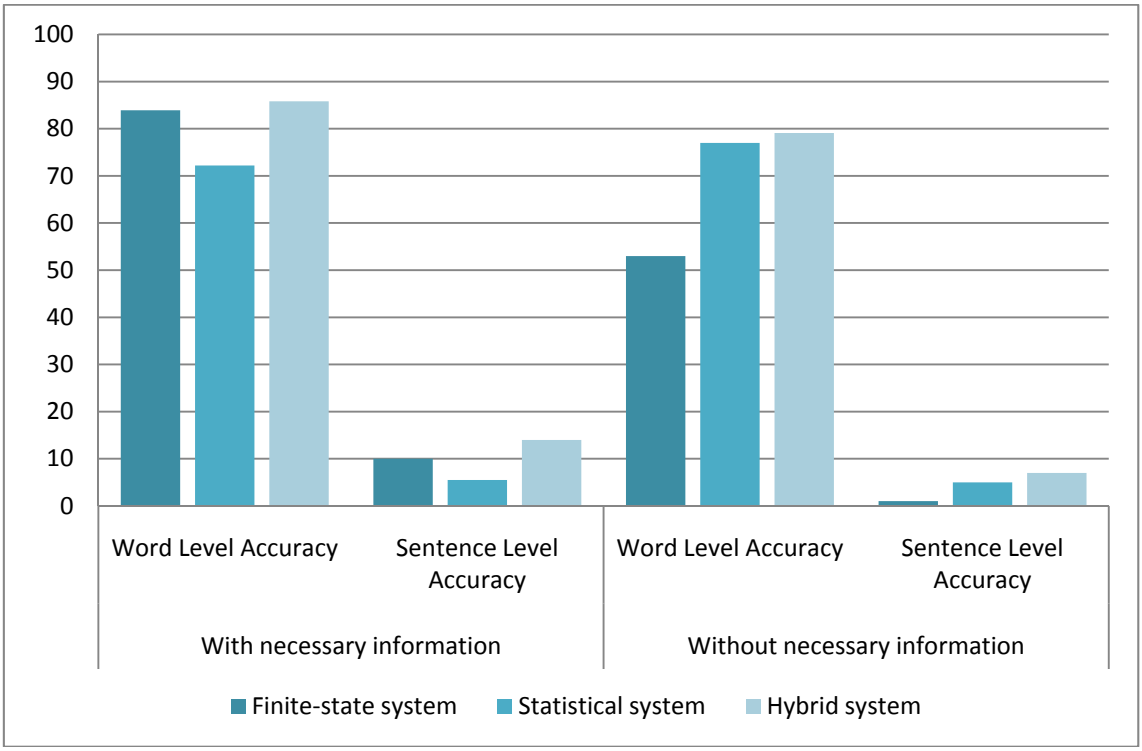


Figure 70: Comparison of Urdu to Hindi scriptural translation systems at word and sentence level

The analysis of our scriptural translation results shows that subjective evaluations like user satisfaction, usability of translation result in real life, fluency and adequacy of translated texts are also very important in addition to the objective evaluations like word accuracy, sentence accuracy, and edit-distance. For subjective evaluation purposes, we have devised different scales to compute different subjective measures for our scriptural translation systems.

To get a real life rating for our translation systems and improve the performance of our scriptural translation systems, we have also developed an online interactive scriptural translation system. This interactive scriptural translation system serves different purposes.

Although we were not yet able to develop practical interdialectal translation systems, we have presented a theoretical study of how we can develop interdialectal translation systems based on Statistical Machine Translation (SMT) and syntactic translation (based on syntactic transfer and linguistic framework) approaches. That includes a study of what kind of linguistic resources (parallel lexicons and corpora) and lingware modules (morphological and syntactic analyzers) are required for building interdialectal translation systems. The availability of interactive scriptural translation systems will play a vital role in developing data resources online using the very large Internet community.

We have mainly experimented on Indo-Pak languages, which represent a large population of the world. The Hindi–Urdu pair alone represents 1,017 million speakers around the globe. Only Chinese has more than 1,000 million speakers. Table 61 shows the number of speakers of Indo-Pak languages.

Sr.	Language	Number of Speakers
1	Hindi	853,000,000
2	Urdu	164,290,000
3	Punjabi	120,000,000
4	Sindhi	21,382,120
5	Seraiki	13,820,000
6	Kashmir	5,640,940
Total		**1178,133,060**

Table 61: Number of speakers of Indo-Pak languages

We have made available online our scriptural translation system for Hindi–Urdu[47], Punjabi/Shahmukhi–Punjabi/Gurmukhi[48] and Seraiki/Shahmukhi–Seraiki/Devanagari[49]. We will also make available scriptural translation systems for Sindhi/Sindhi–Sindhi/Devanagari and Kashmiri/Urdu–Kashmiri/Devanagari in the near future.

We have presented theoretical studies for the development of interdialectal translation in the third part of this thesis. In future, we intend to use use this study and develop necessary linguistic resources and lingware modules for developing statistical and syntactic translation systems for Indo-Pak languages. Table 61 shows the importance of this study in terms of the size of the populations that, we hope, will benefit from our study.

[47] http://puran.info/HUMT/index.html

[48] http://puran.info/PMT/index.html

[49] http://puran.info/saraikiMT/index.html

Bibliography

[1] AbdulJaleel N. and Larkey L. S. 2003. Statistical Transliteration for English-Arabic Cross Language Information Retrieval. In proc. of 12th international Conference on information and Knowledge Management (CIKM 03), New Orleans, pp. 139-146.

[2] Afzal M. and Hussain S. 2001. Urdu Computing Standards: development of Urdu Zabta Takhti (UZT) 1.01. In proc. of IEEE International Multi-Topic Conference, Lahore, pp. 216-222.

[3] Al-Onaizan Y. and Knight K. 2002. Machine Transliteration of Names in Arabic Text. In proc. of Workshop on Computational Approaches To Semitic Languages, the 40th Annual Meeting of the ACL, Philadelphia, Pennsylvania, pp. 1-13.

[4] Al-Onaizan Y. and Knight K. 2002. Translating Named Entities using Monolingual and Bilingual Resources. In proc. of 40th Annual Meeting on ACL, Philadelphia, Pennsylvania, pp. 400-408.

[5] Al-Onaizan Y. and Papineni K. 2006. Distortion Models for Statistical Machine Translation. In proc. of 21st International Conference on Computational Linguistics (COLING) and 44th Annual Meeting of the ACL, Sydney, Australia, pp. 529–536.

[6] Alvarez Benito G. and de Amores Carredano J. G. 1994. Contextual Deletion of Object and Ambiguity in Machine Translation. *Revista alicantina de estudios ingleses*, *7*: pp. 23-36.

[7] Ananthakrishnan R., Bhattacharyya P., Hegde J., Shah R. M. and Sasikumar M. 2008. Simple Syntactic and Morphological Processing Can Help English-Hindi Statistical Machine Translation. In proc. of International Joint Conference on NLP (IJCNLP), Hyderabad.

[8] Ananthakrishnan R. and Rao D. 2003. A Lightweight Stemmer for Hindi. In proc. of Workshop on Computational Linguistics for South-Asian Languages, 10th conference of EACL, Budapest.

[9] Arbabi M., Fischthal S. M., Cheng V. C. and Bart E. 1994. Algorithms for Arabic Name Transliteration. *IBM J. Res. Dev.*, *38* (2): pp. 183-193.

[10] Bar-Hillel Y. 1960. The Present Status of Automatic Translation of Languages. *Advances in Computers*, *1*: pp. 158-163.

[11] Barrachina S., Bender O., Casacuberta F., Civera J., Cubel E., Khadivi S., Lagarda A., Ney H., Tomás J., Vidal E. and Vilar J.-M. 2009. Statistical Approaches to Computer-Assisted Translation. *Computational Linguistics*, *35* (1): pp. 3-28.

[12] Beesley K. R. and Karttunen L. 2003. Finite State Morphology. *CSLI Publications*.

[13] Ben-Ari D., Berry D. M. and Rimon M. 1988. Translational Ambiguity Rephrased. In proc. of 2nd International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language, Carnegie-Mellon University, Pittsburgh, pp. 12-14.

[14] Bhatia T. K. 2003. The Gurmukhi Script and Other Writing Systems of Punjab: History, Structure and Identity. In proc. of International Symposium on Indic Script: Past and Future, pp. 181–213.

[15] Bilac S. and Tanaka H. 2004. A Hybrid Back-Transliteration System for Japanese. In proc. of 20th International Conference on Computational Linguistics (COLING), Geneva, Switzerland.

[16] Bilac S. and Tanaka H. 2004. Improving Back-transliteration by Combining Information Sources. In proc. of International Joint Conference on Natural Language Processing (IJCNLP), Hainan Island, pp. 542-547.

[17] Blanchon H. 1991. Problèmes de désambiguisation interactive et TAO personnelle. In proc. of L'environement traductionnel, journées scientifiques du réseau thématique de recherche Lexicologie, Terminologie, Traduction (LTT), Mons, pp. 31-48.

[18] Blanchon H. 1994. LIDIA-1: une première maquette vers la TA interactive pour tous. Université Joseph Fourier - Grenoble 1, Grenoble, Thèse, 340 p.

[19] Bogel T., Butt M., Hautli A. and Sulger S. 2007. Developing a Finite-State Morphological Analyzer for Urdu and Hindi. In proc. of 6th International Workshop on Finite-State Mehods and Natural Language Processing (FSMNLP), Potsdam.

[20] Bogel T., Butt M. and Sulger S. 2008. Urdu Ezafe and the Morphology-Syntax Interface. In proc. of the LFG conference, Sydney.

[21] Boitet C. 1988. L'apport de Bernard Vauquois à la traduction automatique et au traitement automatique des langues naturelles. In proc. of Colloque sur l'Histoire de l'Informatique (conference on History of Computer Science), France, pp. 63-82.

[22] Boitet C. 1990. Towards Personal MT: general design, dialogue structure, potential role of speech. In proc. of 13th International Conference On Computational Linguistics (COLING), Helsinki, Finland, pp. 30-35.

[23] Boitet C. 1993. Crucial Open Problems in Machine Translation and Interpretation. In proc. of First Symposium on Natural Language Processing, Bangkok, Thailand, pp. 1-29.

[24] Boitet C. 1995. Factors for Success (and Failure) in Machine Translation – some lessons of the first 50 years of R&D. In proc. of MT Summit V, Luxembourg.

[25] Boitet C. 2000. Traduction Assistée par Ordinateur (TAO). in Pierrel ed. *Ingénierie des langues*, Hermès, Paris, pp. 271-292.

[26] Boitet C. 2003. Machine Translation. in Nadel ed. *MacMillan Encyclopedia of Cognitive Science*, MacMillan, London.

[27] Boitet C. 2005. New Architecture for "Democratic" Tuneable Quality MT Systems. In proc. of Conference of the Pacific Association for Computational Linguistics (PACLING-2005), Tokyo, Japan.

[28] Boitet C. 2008. Les architectures linguistiques et computationnelles en traduction automatique sont indépendantes. In proc. of Traitement Automatique des Langues Naturelles (TALN-2008), Avignon, France.

[29] Boitet C. and Gerber R. 1984. Expert Systems and Other New Techniques in MT Systems. In proc. of 10th International Conference on Computational Linguistics (COLING) and the 22nd annual meeting of ACL, Stanford, California, USA, pp. 468-471.

[30] Boitet C. and Nedobejkine N. 1986. Toward Integrated Dictionaries for M(a)T: motivations and linguistic organisation. In proc. of 11th International Conference On Computational Linguistics (COLING), Bonn, Germany, pp. 423-428.

[31] Boitet C. and Tchéou F. X. 1990. On a Phonetic and Structural Encoding of Chinese Characters in Chinese texts. In proc. of ROCLING III, Taipeh, pp. 73-80.

[32] Brown P. F., Cocke J., Pietra S. A. D., Pietra V. J. D., Jelinek F., Lafferty J. D., Mercer R. L. and Roossin P. S. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, *16* (2): pp. 79-85.

[33] Brown P. F., Pietra S. A. D., Pietra V. J. D. and Mercer R. L. 1993. The Mathematics of Statistical Machine Translation: parameter estimation. *Computational Linguistics*, *19* (2): pp. 263-312.

[34] Brown R. D. and Nirenburg S. 1990. Human-Computer Interaction for Semantic Disambiguiation. In proc. of 13th International Conference on Computational Linguistics (COLING), Helsinki, Finland, pp. 42-47.

[35] Butt M., Dyvik H., King T. H., Masuichi H. and Rohrer C. 2002. The Parallel Grammar Project. In proc. of workshop on Grammar Engineering and Evaluation, the 19th International Conference on Computational Linguistics (COLING), Taipei, Taiwan, pp. 1-7.

[36] Butt M., Forst M., King T. H. and Kuhn J. 2003. The Feature Space in Parallel Grammar Writing. In proc. of ESSLLI 2003 Workshop on Ideas and Strategies for Multilingual Grammar Development, Vienna, Austria, pp. 9-16.

[37] Butt M. and King T. H. 2007. Urdu in a Parallel Grammar Development Environment. *Language Resources and Evaluation, Special Issue on Asian Language Processing: State of the Art Resources and Processing*, *41*: pp. 191-207.

[38] Butt M., King T. H. and Maxwell J. 2003. Productive Encoding of Urdu Complex Predicates in the ParGram Project. In proc. of workshop on Computational Linguistics for South Asian Languages: Expanding Synergies with Europe, 10th Conference of EACL, Budapest, Hungary, pp. 9-13.

[39] Butt M., King T. H., Nino M.-E. and Segond F. 1999. A Grammar Writer's Cookbook. *Stanford: CSLI Publications*.

[40] Callison-Burch C., Osborne M. and Koehn P. 2006. Re-evaluating the Role of BLEU in Machine Translation Research. In proc. of 11th Conference of EACL, Trento, Italy, pp. 249-256.

[41] Carl M. and Way A. (eds.). 2003. Recent Advances in Example-Based Machine Translation. *Kluwer Acadmic Publishers*.

[42] Carlos C. S., Choudhury M. and Dandapat S. 2009. Large-coverage Root Lexicon Extraction for Hindi. In proc. of 12th conference of EACL, Athens, pp. 121-129.

[43] Chan Y. S. and Ng H. T. 2008. MAXSIM: a maximum similarity metric for Machine Translation Evaluation. In proc. of 46th Annual Meeting of the ACL, Human Language Technology (HLT), Columbus, Ohio, pp. 55-62.

[44] Charniak E. 1996. Tree Bank Grammars, Department of Computer Science, Brown University, Providence, pp. 1-12.

[45] Chaudhary S. 1987. Heer Waris Shah. *Lehran (Punjabi Jouranl in Punjabi/Shahmukhi)*.

[46] Chinnakotla M. K. and Damani O. P. 2009. Experiences with English-Hindi, English-Tamil and English-Kannada Transliteration Tasks at NEWS 2009. In proc. of workshop on Named Entities (NEWS-09), Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing ACL/IJCNLP Singapore, pp. 44-47.

[47] Deselaers T., Hasan S., Bender O. and Ney H. 2009. A Deep Learning Approach to Machine Transliteration. In proc. of 4th Workshop on Statistical Machine Translation, the 12th Conference of EACL, Athens, pp. 233-241.

[48] Din F. u. فیروزاللغات اُردُو (Urdu Dictionary). *Feroz Sons Publishers*, Lahore.

[49] Doddington G. 2002. Automatic Evaluation of Machine Translation Quality using N-gram Co-occurrence Statistics. In proc. of 2nd International Conference on Human Language Technology Research, San Diego, California, USA, pp. 138-145.

[50] Ekbal A., Naskar S. K. and Bandyopadhyay S. 2006. A Modified Joint Source-channel Model for Transliteration. In proc. of 21st International Conference on Computational Linguistics (COLING) and the 44th Annual Meeting of the ACL, Sydney, pp. 191-198.

[51] Emele M. C. and Dorna M. 1998. Ambiguity Preserving Machine Translation using Packed Representations. In proc. of 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics (COLING), Montreal, Quebec, Canada, pp. 365-371.

[52] Finch A. and Sumita E. 2009. Transliteration by Bidirectional Statistical Machine Translation. In proc. of workshop on Named Entities (NEWS-09), Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing ACL/IJCNLP, Singapore, pp. 52-56.

[53] Foster G. 2002. Text Prediction for Translators. Université de Montréal, Montréal, Canada, PhD.

[54] Foster G., Isabelle P. and Plamondon P. 1996. Word Completion: a first step toward target-text mediated IMT. In proc. of 16th International Conference on Computational Linguistics (COLING), Copenhagen, Denmark, pp. 394-399.

[55] Foster G., Isabelle P. and Plamondon P. 1998. Target-Text Mediated Interactive Machine Translation. *Machine Translation*, *12* (1/2): pp. 175-194.

[56] Fraser A. and Marcu D. 2007. Measuring Word Alignment Quality for Statistical Machine Translation. *Computational Linguistics*, *33* (3): pp. 293-303.

[57] Fujii A. and Ishikawa T. 2001. Japanese/English Cross-Language Information Retrieval: exploration of query translation and transliteration. *Computers and the Humanities*, *35* (4): pp. 389-420.

[58] Ganchev K., Graça J. V. and Taskar B. 2008. Better Alignments = Better Translations? In proc. of 46th Annual Meeting of the ACL, Human Language Technology (HLT), Columbus, Ohio, pp. 986-993.

[59] Gao W., Wong K. F. and Lam W. 2004. Phoneme-based Transliteration of Foreign Names for OOV Problem. In proc. of 1st International Joint Conference on Natural Language Processing (IJCNLP), Hainan Island, pp. 110-119.

[60] Goto I., Kato N., Ehara T. and Tanaka H. 2004. Back Transliteration from Japanese to English using Target English Context. In proc. of 20th International Conference on Computational Linguistics (COLING), Geneva.

[61] Goto I., Kato N., Uratani N. and Ehara T. 2003. Transliteration Considering Context Information based on the Maximum Entropy Method. In proc. of MT Summit IX, New Orleans.

[62] Graehl J. and knight K. 2004. Training Tree Transducers. In proc. of Human Lanuage Technology (HLT) and annual meeting of NAACL, Boston, pp. 105-112.

[63] Graehl J., knight K. and May J. 2008. Training Tree Transducers. *Computational Linguistics*, *34* (3): pp. 391-427.

[64] Hagège C. 1982. La structure des langues. *PUF*, Paris.

[65] Haizhou L., Min Z. and Jian S. 2004. A Joint Source-channel Model for Machine Transliteration. In proc. of 42nd Annual Meeting on ACL, Barcelona.

[66] Halpern J. 2002. Lexicon-based Orthographic Disambiguation in CJK Intelligent Information Retrieval. In proc. of 3rd workshop on Asian language resources and

international standardization, the 19th International Conference on Computational Linguistics (COLING), Taipei, Taiwan, pp. 1-7.

[67] Halpern J. 2010. Orthographic Variation in Chinese, The CJK Dictionary Institute, Inc.

[68] Haque R., Dandapat S., Srivastava A. K., Naskar S. K. and Way A. 2009. English-Hindi Transliteration Using Context-Informed PB-SMT: the DCU system for NEWS 2009. In proc. of workshop on Named Entities (NEWS-09), Joint conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing ACL/IJCNLP, Singapore, pp. 104-107.

[69] Hieronymus J. 1993. ASCII Phonetic Symbols for the World's Languages: Worldbet, AT&T Bell Laboratories.

[70] Holmqvist M., Stymne S., Foo J. and Ahrenberg L. 2009. Improving Alignments for SMT by Reordering and Augmenting the Training Corpus. In proc. of 4th workshop on Statistical Machine Translation, the 12th Conference of EACL, Athens, Greece, pp. 120-124.

[71] Hong G., Kim M.-J., Lee D.-G. and Rim H.-C. 2009. A Hybrid Approach to English-Korean Name Transliteration. In proc. of Workshop on Named Entities (NEWS-09), Joint conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing ACL/IJCNLP Singapore, pp. 108-111.

[72] Hovy E. H. 1999. Toward Finely Differentiated Evaluation Metrics for Machine Translation. In proc. of EAGLES Workshop on Standards and Evaluation, Pisa, Italy, pp. 127-133.

[73] Huang F. 2005. Cluster-specific Named Entity Transliteration. In proc. of Conference on Human Language Technology and Empirical Methods in Natural Language Processing, Vancouver, pp. 435-442.

[74] Huang F. 2009. Confidence Measure for Word Alignment. In proc. of Joint Conference the 47th Annual Meeting of the ACL and the 4th IJCNLP, Singapore, pp. 932-940.

[75] Humayoun M. 2006. Urdu Morphology, Orthography and Lexicon Extraction. Chalmers University of Technology, Master, 108 p.

[76] Humayoun M., Hammarström H. and Ranta A. 2007. Urdu Morphology, Orthography and Lexicon Extraction. In proc. of 2nd workshop on computational approaches to Arabic script-based languages, Stanford University.

[77] Hussain S. 2004. Finite-State Morphological Analyzer for Urdu. National University of Computer and Emerging Sciences, Lahore, Master, 197 p.

[78] Hussain S. 2004. Letter-to-Sound Conversion for Urdu Text-to-Speech System. In proc. of Workshop on Computational Approaches to Arabic Scriptbased Languages, International Conference on Computational Linguistics (COLING), Geneva, pp. 74-79.

[79] Hussain S. and Afzal M. 2001. Urdu Computing Standards: Urdu Zabta Takhti (UZT) 1.01. In proc. of IEEE International Multi-Topic Conference, Lahore, pp. 223-228.

[80] Hutchins W. J. 1986. Machine Translation: past, present, future. *Ellis Horwood Ltd*.

[81] Huynh C.-P., Boitet C. and Blanchon H. 2008. SECTra_w.1: an online collaborative system for evaluating, post-editing and presenting MT translation corpora. In proc. of 6th International Language Resources and Evaluation (LREC), Marrakech, Morocco, pp. 2571-2576.

[82] Jeong K. S., Myaeng S. H., Lee J. S. and Choi K.-S. 1999. Automatic Identification and Back-transliteration of Foreign Words for Information Retrieval. *Information Processing and Management*, *35*: pp. 523-540.

[83] Jung S. Y., Hong S. L. and Paek E. 2000. An English to Korean Transliteration Model of Extended Markov Window. In proc. of 18th International Conference On Computational Linguistics (COLING), Saarbrücken, pp. 383-389.

[84] Kang B. and Choi K. 2000. Automatic Transliteration and Back Transliteration by Decision Tree Learning. In proc. of 2nd International Conference on Evaluation and Language Resources (ELRC), Athens.

[85] Kang I. and Kim G. 2000. English-to-Korean Transliteration using Multiple Unbounded Overlapping Phoneme Chunks. In proc. of 18th Conference on Computational Linguistics, Saarbrücken.

[86] Kanthak S. and Ney H. 2004. FSA: an efficient and flexible C++ toolkit for finite state automata using on-demand computation. In proc. of 42nd Annual Meeting on Association for Computational Linguistics, Barcelona.

[87] Kaplan R. M. and Kay M. 1994. Regular Models of Phonological Rule Systems. *20* (3).

[88] Karlsson F., Voutilainen A., Heikkila J. and Anttila A. 1995. Constraint Grammar, a Language Independent System for Parsing Unrestricted Text. *Mouton de Gruyter*.

[89] Karttunen L. 1997. The Replace Operator. in *Finite-state Language Processing*, MIT Press, Cambridge.

[90] Karttunen L., Kaplan R. M. and Zaenen A. 1992. Two-level Morphology with Composition. In proc. of 14th International Conference on Computational Linguistics (COLING), Nantes.

[91] Kay M. 1973. The MIND System. in Rustin ed. *Natural Language Processing*, Algorithmic Press, New York, pp. 155-188.

[92] Kay M. 1997. The Proper Place of Men and Machines in Language Translation. *Machine Translation*, *12* (1/2): pp. 3-23.

[93] Kay M., Gawron J. M. and Norvig P. 1994. Verbmobil: a translation system for face-to-face dialog. *CSLI Lecture Notes*.

[94] Kellogg R. S. H. 1872. A Grammar of Hindi Language. *Oriental Book Reprint*, Delhi.

[95] Khan M. A. 1997. اُردو کا صوتی نظام (Sound System in Urdu). *National Language Authority*, Islamabad.

[96] King T. H., Forst M., Kuhn J. and Butt M. 2005. The Feature Space in Parallel Grammar Writing *Journal of Research on Language and Computation; Special Issue on "Shared Representations in Multilingual Grammar Engineering*, Kluwer Academic Press.

[97] Kirschenbaum A. and Wintner S. 2009. Lightly Supervised Transliteration for Machine Translation. In proc. of 12th Conference of the European Chanpter of the ACL, Athens, pp. 433-441.

[98] Kleene S. C. 1956. Representation of Events in Nerve Nets and Finite Automata. *Automata Studies*.

[99] Knight K. and Al-Onaizan Y. 1998. Translation with Finite-State Devices In proc. of 3rd Conference of the Association for Machine Translation in the Americas on Machine Translation and the Information Soup (AMTA-98), Pennsylvania, pp. 421-437.

[100] Knight K. and Graehl J. 1997. Machine Transliteration. In proc. of 8th Conference of EACL, Madrid, pp. 128-135.

[101] Knight K. and Graehl J. 1998. Machine Transliteration. *Computational Linguistics*, *24* (4): pp. 599-612.

[102] Koehn P. 2010. Statistiacal Machine Translation. *Cambridge University Press*, pp. 446.

[103] Koehn P. and Haddow B. 2009. Interactive Assistance to Human Translators using Statistical Machine Translation Methods. In proc. of MT Summit XII, Ontario, Canada.

[104] Koehn P. and Hoang H. 2007. Factored Translation Models. In proc. of Joint Conference on Empirical Methods in Natural Language Processing (EMNLP) and Computational Natural Language Learning, Prague, pp. 868-876.

[105] Koehn P., Hoang H., Birch A., Callison-Burch C., Federico M., Bertoldi N., Cowan B., Shen W., Moran C., Zens R., Dyer C., Bojar O., Constantin A. and Herbst E. 2007. Moses: open source toolkit for Statistical Machine Translation. In proc. of 47th Association for Computational Linguistics 2007 Demo and Poster Sessions, pp. 177-180.

[106] Koehn P., Och F. J. and Marcu D. 2003. Statistical Phrase-Based Translation. In proc. of Human Language Technology (HLT), NAACL-2003, Edmonton, Canada, pp. 48-54.

[107] Kumar S., Och F. and Macherey W. 2007. Improving Word Alignment with Bridge Languages. In proc. of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 42-50.

[108] Kumaran A. and Kellner T. 2007. A Generic Framework for Machine Transliteration. In proc. of SIGIR 2007, Amsterdam.

[109] Langlais P., Foster G. and Lapalme G. 2000. TransType: a computer-aided translation typing system. In proc. of Workshop on Embedded Machine Translation Systems, ANLP-NAACL, Seattle, Washington pp. 46-51.

[110] Langlais P., Lapalme G. and Loranger M. 2002. TransType: development-evaluation cycles to boost translator's productivity. *Machine Translation*, *15* (4): pp. 77-98.

[111] Lavie A. and Denkowski M. J. 2009. The Meteor Metric for Automatic Evaluation of Machine Translation. *Machine Translation*, *23* (4).

[112] Lee J. S. and Choi. K. S. 1998. English to Korean Statistical Transliteration for Information Retrieval. *Computer Processing of Oriental languages*, *12* (1): pp. 17-37.

[113] Lee Y.-S., Weinstein C., Seneff S. and Tummala D. 1997. Ambiguity Resolution for Machine Translation of Telegraphic Messages. In proc. of 8th conference on European chapter of the Association for Computational Linguistics, Madrid, Spain pp. 120-127.

[114] Lewis M. P. 2009. Ethnologue: Languages of the World. Lewis ed., SIL International, Dallas. http://www.ethnologue.com/web.asp.

[115] Lopez A. 2008. Statistical Machine Translation. *ACM Computing Surveys*, *40* (3).

[116] Ma Y., Ozdowska S., Sun Y. and Way A. 2008. Improving Word Alignment Using Syntactic Dependencies. In proc. of 2nd ACL Workshop on Syntax and Structure in Statistical Translation (SSST-2), the 46th Annual Meeting of the ACL, Columbus, Ohio, pp. 69-77.

[117] Malik M. G. A. 2005. Towards a Unicode Compatible Punjabi Character Set. In proc. of 27th Internationalization and Unicode Conference, Berlin.

[118] Malik M. G. A. 2006. Hindi Urdu Machine Transliteration System, Dept. of Linguistics, University of Paris 7, Paris, pp. 97.

[119] Malik M. G. A. 2006. Punjabi Machine Transliteration. In proc. of 21st international Conference on Computational Linguistics (COLING) and the 44th Annual Meeting of the ACL, Sydney, pp. 1137-1144.

[120] Malik M. G. A., Besacier L., Boitet C. and Bhattacharyya P. 2009. A Hybrid Model for Urdu Hindi Transliteration. In proc. of Joint conference of the 47th Annual Meeting of the Association of Computational Linguistics and the 4th International Joint Conference

on Natural Language Processing of the Asian Federation of NLP ACL/IJCNLP Workshop on Named Entities (NEWS-09), Singapore, pp. 177–185.

[121] Malik M. G. A., Boitet C. and Bhattacharyya P. 2008. Hindi Urdu Machine Transliteration using Finite-state Transducers. In proc. of 22nd International Conference on Computational Linguistics (COLING), Manchester, pp. 537-544.

[122] Malik M. G. A., Boitet C. and Bhattacharyya P. 2010. Analysis of Noori Nast'aleeq for Major Pakistani Languages. In proc. of 2nd Workshop on Spoken Language Technologies for Under-resourced Languages SLTU-2010, Penang, Malaysia.

[123] Maruyama H. and Watanabe H. 1990. An Interactive Japanese Parser for Machine Translation. In proc. of 13th International Conference on Computational Linguistics (COLING), Helsinki, Finland, pp. 257-262.

[124] Matthews D. 2007. Machine transliteration of proper names, University of Edinburgh.

[125] Mel'cuk I. A. 1963. Machine Translation and Linguistics. in O.S.Akhmanova, I.A.Mel'chuk, R.M.Frumkina and E.V.Paducheva eds. *Exact methods in linguistic research*, University of California Press, pp. 44-79.

[126] Melby A. K. 1978. Design and Implementation of a Computer-Assisted Translation System. In proc. of 7th International Conference On Computational Linguistics (COLING), Bergen, Norway, pp. 14-18.

[127] Melby A. K. 1982. Multi-level Translation Aids in a Distributed System. In proc. of 9th International Conference On Computational Linguistics (COLING), Prague, Czechoslovakia, pp. 215-220.

[128] Melby A. K., Smith M. R. and Peterson J. 1980. ITS: INTERACTIVE TRANSLATION SYSTEM. In proc. of 8th International Conference On Computational Linguistics (COLING), Tokyo, Japan, pp. 424-429.

[129] Meng H., Lo W., Chen B. and Tang K. 2001. Generating Phonetic Cognates to Handle Named Entities in English-Chinese Cross-language Spoken Document Retrieval. In proc. of IEEE workshop on Automatic Speech Recognition and Understanding ASUR, Madonna di Campiglio Trento, pp. 311-314.

[130] Min Z., Haizhou L. and Jian S. 2004. Direct Orthographical Mapping for Machine Transliteration. In proc. of 20th International Conference on Computational Linguistics (COLING), Geneva.

[131] Mohri M. 1994. Compact Representation by Finite-state Transducers. In proc. of 32nd Annual Meeting of the Association of Computational Linguistics, Las Cruces.

[132] Mohri M. 1994. Minimization of Sequential Transducers. *LNCS, Springer*.

[133] Mohri M. 1994. Syntactic Analysis by Local Grammars Automata: an efficient algorithm. In proc. of International Conference on Computational Complexity (COMPLEX-94), Budapest.

[134] Mohri M. 1996. On Some Applications of Finite-state Automata Theory to Natural Language Processing. *2* (1).

[135] Mohri M. 1997. Finite-state Transducers in Language and Speech Processing. *23* (2).

[136] Montaut A. 2004. A Linguistic Grammar of Hindi. *Studies in Indo-European Linguistics Series*.

[137] Nabende P. 2009. Transliteration System using pair HMM with Weighted FSTs. In proc. of Joint conference of the 47th Annual Meeting of the Association of Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of NLP ACL/IJCNLP Workshop on Named Entities (NEWS-09), Singapore, pp. 100–103.

[138] Nieben S., Och F. J., Leusch G. and Ney H. 2000. An Evaluation Tool for Machine Translation: fast evaluation for MT Research. In proc. of 2nd International Conference on Language Resources and Evaluation, Athens, Greece.

[139] Nirenburg S., Somers H. and Wilks Y. (eds.). 2003. Readings in Machine Translation. *MIT Press Cambridge*, London.

[140] Noeman S. 2009. Language Independent Transliteration System using Phrase based SMT Approach on Substrings. In proc. of Joint conference of the 47th Annual Meeting of the Association of Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of NLP ACL/IJCNLP Workshop on Named Entities (NEWS-09), Singapore, pp. 112-115.

[141] Och F. J. and Ney H. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, *29* (1): pp. 19-51.

[142] Och F. J. and Ney H. 2004. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, *30* (4): pp. 417-449.

[143] Oh J.-H. and Choi K.-S. 2002. An English-Korean Transliteration Model using Pronunciation and Contextual Rules. In proc. of 19th International Conference on Computational Linguistics (COLING), Taipei.

[144] Oh J.-H. and Choi K.-S. 2005. An Ensemble of Grapheme and Phoneme for Machine Transliteration. In proc. of 2nd International Joint Conference on Natural Language Processing, Jeju Island, pp. 450-461.

[145] Oh J.-H., Choi K.-S. and Isahara H. 2006. A Comparison of Different Machine Transliteration Models. *27*: pp. 119-151.

[146] Oh J.-H., Choi K.-S. and Isahara H. 2006. A Machine Transliteration Model based on Correspondence between Graphemes and Phonemes. *ACM Transactions on Asian Language Information Processing (TALIP)*, *5* (3).

[147] Papineni K., Roukos S., Ward T. and Zhu W.-j. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In proc. of 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, pp. 311-318.

[148] Paul M., Finch A. and Sumita E. 2009. NICT@WMT09: Model Adaptation and Transliteration for Spanish-English SMT. In proc. of 4th Workshop on Statistical Machine Translation, the 12th Conference of EACL, Athens, Greece, pp. 105-109.

[149] Pericliev V. 1984. Handling Syntactical Ambiguity In Machine Translation. In proc. of 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics (COLING/ACL), Stanford University, California, USA, pp. 521-524.

[150] Pirkola A., Toivonen J., Keskustalo H., Visala K. and Järvelin K. 2003. Fuzzy Translation of Cross-lingual Spelling Variants. In proc. of 26th Annual international ACM SIGIR Conference on Research and Development in informaion Retrieval, Toronto.

[151] Platts J. T. 1884. A Dictionary of Urdu, Classical Hindi and Englsih, W. H. Allen & Co., London.

[152] Platts J. T. 1909. A Grammar of the Hindustani or Urdu Language. *Crosby Lockwood and Son*.

[153] Qu Y., Grefenstette G. and Evans D. A. 2003. Automatic Transliteration for Japanese-to-English Text Retrieval. In proc. of 26th annual international ACM SIGIR conference on Research and development in information retrieval, Toronto, pp. 353-360.

[154] Qu Y., Grefenstette G. and Evans D. A. 2003. Resolving Translation Ambiguity Using Monolingual Corpora *Lecture Notes in Computer Science*, *2785/2003* (0302-9743 (Print) 1611-3349 (Online)): pp. 223-241.

[155] Rahman T. 1997. Language and Politics in Pakistan. *Oxford University Press*, Lahore.

[156] Rahman T. 2004. Language Policy and Localization in Pakistan: Proposal for a Paradigmatic Shift. In proc. of Crossing the Digital Divide, SCALLA Conference on Computational Linguistics, Katmandu.

[157] Rai A. 2000. Hindi Nationalism. *Orient Longman Private Limited*, New Delhi.

[158] Ramanathan A., Choudhary H., Ghosh A. and Bhattacharyya P. 2009. Case Markers and Morphology: Addressing the crux of the Fluency Problem in English-Hindi SMT. In proc. of 47th annual meeting of ACL and 4th International Joint Conference on NLP (IJCNLP), Singapore, pp. 800-808.

[159] Rauf S. A. 2006. Urdu Morphology - Adjectives and Adverbs. University of Paris 7, Paris, Master.

[160] Ray P. R., Harish V., Basu A. and Sarkar S. 2003. Part of Speech Tagging and Local Word Grouping Techniques for Natural Language Parsing in Hindi. In proc. of ICON, Maysor.

[161] Roche E. 1993. Analyse syntaxique transformationnelle du français par transducteurs et lexique-grammaire, Paris.

[162] Roche E. and Schabes Y. (eds.). 1997. Finite-state Language Processing. *MIT Press*, Cambridge.

[163] Rounds W. C. 1970. Mappings and Grammars on Trees. *Mathematical Systems Theory*, *4* (3): pp. 257-287.

[164] Saini T. S. and Lehal G. S. 2008. Shahmukhi to Gurmukhi Transliteration System: A Corpus based Approach. *Research in Computing Science*, *33*: pp. 151-162.

[165] Sakai T., Kumano A. and Manabe T. 2002. Generating Transliteration Rules for Cross-language Information Retrieval from Machine Translation Dictionaries. In proc. of IEEE Conference on Systems, Man and Cybernatics.

[166] Schützenberger M. P. 1976. Sur les relations rationnelles enter monoïdes libres. *3*: pp. 243-259.

[167] Schützenberger M. P. 1977. Sur une variante des fonctions sequentielles. *4*: pp. 47-57.

[168] Sherif T. and Kondrak G. 2007. Substring-based Transliteration. In proc. of 45th Annual Meeting of the ACL, Prague.

[169] Silberztein M. 1993. Dictionnaires électroniques et analyse automatique de textes: le système INTEX. *Masson*, Paris.

[170] Singh S., Gupta K., Shrivastava M. and Bhattacharyya P. 2006. Morphological Richness Offsets Resource Demand-Experiences in Constructing a POS Tagger for Hindi. In proc. of 21st international Conference on Computational Linguistics (COLING) and the 44th Annual Meeting of the ACL, Sydney, pp. 779-786.

[171] Slocum J. 1984. Machine Translation: its history, current status, and future prospects. In proc. of 22nd Annual Meeting of the Association for Computational Linguistics and the 10th International Conference on Computational Linguistics (COLING), Stanford University, California, pp. 546-561.

[172] Slocum J. 1985. A survey of Machine Translation: its history, current status, and future prospects. *Computational Linguistics*, *11* (1): pp. 1-17.

[173] Song Y., Kit C. and Chen X. 2009. Transliteration of Name Entity via Improved Statistical Translation on Character Sequences. In proc. of Joint conference of the 47th Annual Meeting of the Association of Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of NLP ACL/IJCNLP Workshop on Named Entities (NEWS-09), Singapore, pp. 57-60.

[174] Stall B. and Knight K. 1998. Translating Names and Technical Terms in Arabic Text. In proc. of Workshop on Computational Approaches to Semitic Languages, COLING/ACL, Montreal, pp. 34-41.

[175] Stolcke A. 2002. SRILM - An Extensible Language Modeling Toolkit. In proc. of International Conference on Spoken Language Processing, pp. 901-904.

[176] Thatcher J. W. 1970. Generalized Sequential Machine Maps. *Journal of Computer and System Sciences*, *4*: pp. 339-367.

[177] Turian J. P., Shen L. and Melamed I. D. 2003. Evaluation of Machine Translation and its Evaluation. In proc. of MT Summit IX, New Orleans, U.S.A.

[178] Vauquois B. 1976. Automatic Translation – a survey of different approaches. In proc. of International Conference on Computational Linguistics (COLING), pp. 127-135.

[179] Vauquois B. and Boitet C. 1985. Automated Translation at Grenoble University. *Computational Linguistics*, *11* (1): pp. 28-36.

[180] Virga P. and Khudanpur S. 2003. Transliteration of Proper Names in Cross-language Applications. In proc. of 26th Annual international ACM SIGIR Conference on Research and Development in informaion Retrieval, Toronto.

[181] Virga P. and Khudanpur S. 2003. Transliteration of Proper Names in Cross-lingual Information Retrieval. In proc. of ACL 2003 Workshop on Multilingual and Mixed-Language Named Entity Recognition, Sapporo.

[182] Weaver W. (ed.), 1955. Translation. *MIT Press*, Cambridge, MA.15-23 p.

[183] Wehrli E. and Ramluckun M. 1993. ITS-2 : an interactive personal translation system. In proc. of 6th Conference of EACL, Utrecht, Netherlands pp. 476-477.

[184] Wells J. C. 1995. Computer-coding the IPA: a proposed extension of SAMPA *www.phon.ucl.ac.uk/home/sampa*, University College London.

[185] White J. S., O'Connell T. and O'Mara F. 1994. The ARPA MT Evaluation Methodologies: evaluation, lessons and future approaches. In proc. of 1st Conference of the Association for Machine Translation in America, Columbia, USA, pp. 193-205.

[186] Whitelock P. J., Wood M. M., Chandler B. J., Holden N. and Horsfall H. J. 1986. Strategies for Interactive Machine Translation: the experience and implementations of the UMIST Japanese project. In proc. of 11th International Conference on Compuational Linguistics (COLING), Bonn, West Germany, pp. 329-334.

[187] Wilks Y. 2009. Machine Translation: Its scope and limits. *Springer*.

[188] Woods W. A. 1970. Transition Network Grammars for Natural Language Analysis. *13* (10).

[189] Yan Q., Gregory G. and David A. E. 2003. Automatic Transliteration for Japanese-to-English Text Retrieval. In proc. of 26th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 353-360.

[190] Yngve V. H. 1957. A Framework for Syntactic Translation. *Mechanical Translation*, *4* (3): pp. 59-65.

[191] Yoon S.-Y., Yoon S.-Y. and Yoon S.-Y. 2007. Multilingual Transliteration using Feature based Phonetic Method. In proc. of 45th Annual Meeting of the ACL, Prague.

[192] Zajac. R. 1988. Interactive Translation : a new approach. In proc. of 12th International Conference On Computational Linguistics (COLING), Budapest, Hungary, pp. 785-790.

[193] Zens R. and Ney H. 2003. A Comparative Study on Reordering Constraints in Statistical Machine Translation. In proc. of 41st Annual Meeting on Association for Computational Linguistics, Sapporo, Japan, pp. 144-151.

[194] Zia K. 1999. Standard Code Table for Urdu. In proc. of 4th Symposium on Multilingual Information Processing (MLIT-4), Yangon.

[195] Zia K. 1999. Towards Unicode Standard for Urdu. In proc. of 4th Symposium on Multilingual Information Processing (MLIT-4), Yangon.

Annexes

Annex 1 Analysis of Writing Systems: Urdu, Punjabi/Shahmukhi, Seraiki/Shahmukhi and Kashmiri

Analysis of Urdu

Pakistan is a country with at least 6 major languages and 58 minor ones. The national language, Urdu, has over 11 million mother-tongue speakers, while those who use it as a second language could well be more than 105 millions. It is also spoken as a mother tongue by over 48 million persons in India as well as by the diaspora settled in the Arab states (mainly the Gulf), Mauritius, Britain, North America and the rest of the world (estimated to be over 60 millions) [156]. It also enjoys the status of being one of the official languages of India and is also the official language of different states of India like Uttar Pradesh, Indian-controlled Jammu and Kashmir, etc. The total count of persons who can speak and understand Urdu is more than 160 million [156].

Brief History of Urdu

Urdu belongs to the Indo-Aryan family and thus is an Indo-European language. It has developed to its present shape under the strong influence of Arabic, Persian, Punjabi, Sanskrit, and other indigenous languages of the Indian sub-continent during the Delhi Sultanate and the Mughal Empire from the 12th to the 18th century.

Muslims first started directly to influence the Indian sub-continent by the start of the 8th century, when in 712 Muhammad Bin Qasim, a 17-year old General, attacked Daibul (now Karachi, Pakistan) to rescue Muslim women and children from the prison of Raja Dahir, ruler of Sindh (a province in south west of Pakistan). After that there is a long chain of Muslim conquerors like Mahmud Ghaznavi, Shahab-ud-din Muhammad Ghuri, Zaheerud-din Muhammad Babar, etc. The Delhi Sultanate was established by Sahab-ud-din Muhammad Ghuri and was managed by his successors like Qutab-ud-din Aibak, Shams-uddin Altutmush, etc. from the 13th to the 16th century. This was the first Muslim empire in this area. In the 16th century, Zahir-ud-din Muhammad Babur, the first Mughal emperor, established the Mughal Empire in the Indian sub-continent. The Mughal Empire continued from 1526 to 1857.

The kings of the Delhi Sultanate and the Mughal Empire brought the Persian language with them. Persian, being the official language of the king's court, was considered the language of power. So it had a strong influence on the indigenous languages. Also these kings had brought large armies with them who could speak Arabic, Persian and Turkish. Urdu started to develop by the interaction of invading army men and indigenous people during the Delhi Sultanate and the Mughal Empire. The Persian language played an important and crucial role in the formation and development of common languages of the central, north and northwest regions of south Asia. Following the vast Islamic empire in south Asia, a hybrid language of Arabic, Persian, Turkish, and indigenous languages began to form around the 10th and the 11th century. One dialect of this hybrid language would eventually be known as Urdu. Urdu is a Turkish word meaning 'army' or 'herd'. It grew from the interaction of (often Persian-speaking) Muslim soldiers and native people. Soon the Persian script and Nasta'leeq form of cursive style were adopted, with additional characters added to accommodate the South-Asian phonological sys-

tem. A new language, based on the South-Asian grammar with a vocabulary largely divided between Persian and indigenous languages, developed and was called Urdu.

Urdu soon gained distinction as the preferred language in the Persian courts of the Muslim Empire and to this day retains an important place in the literary and cultural spheres. Many distinctly Persian forms of literature, such as 'Ghazals' and 'Nazms', came to influence and also to be affected by the South-Asian culture, producing a distinct melding of Middle-Eastern and South-Asian heritages. A famous cross-over writer was Amir Khusro whose Persian and Urdu couplets are read to this day in the subcontinent.

Urdu Alphabet

Urdu is transcribed in a derivation of the Persian alphabet that is itself a derivation of the Arabic alphabet. It is read and written from right to left. Nasta'leeq, a cursive, context-sensitive and a highly complex writing system, is widely used for the Urdu calligraphy. The shape assumed by a character in a word is context-sensitive. The Urdu alphabet contains 35 simple consonants, 15 aspirated consonants, one character for nasal sound, 15 diacritical marks, 10 digits and other symbols. In Unicode, Arabic and its associated languages like Persian, Urdu, Punjabi, Sindhi etc. have been allocated 1,200 code points (0600h – 06FFh, FB50h – FEFFh) [195].

Urdu Alphabet Standard

The Urdu standard regulating body in Pakistan is National Language Authority (hereafter NLA). NLA has developed the Urdu Zabta Takhti (UZT 1.01 – Standard Urdu Alphabet) [2, 79] that is given in Figure 71. Unicode values and IPAs (International Phonetic Alphabet) are also given with each character in Figure 71 to make it compatible with Unicode standards and to make it understandable by unacquainted readers.

It is a 256 bit codepage and has been divided into various logical sections. It is divided into the following logical sections:

1. Control Characters (0 – 31, 127)
2. Punctuation and Arithmetic Symbols (32 – 47, 58 – 65)
3. Digits (48 – 57)
4. Urdu Diacritics (66 – 79, 123 – 126)
5. Urdu Characters (80 – 122)
6. Reserved control characters (128 – 159, 255)
7. Special Symbols (160 – 176, 192 – 199)
8. Reserved expansion Space (177 – 191, 200 – 207, 240 – 253)
9. Vendor Area (208 – 239)
10. Toggle character (254)

The figure below shows the UZT [2, 79], approved by the National Language Authority (NLA) of Pakistan.

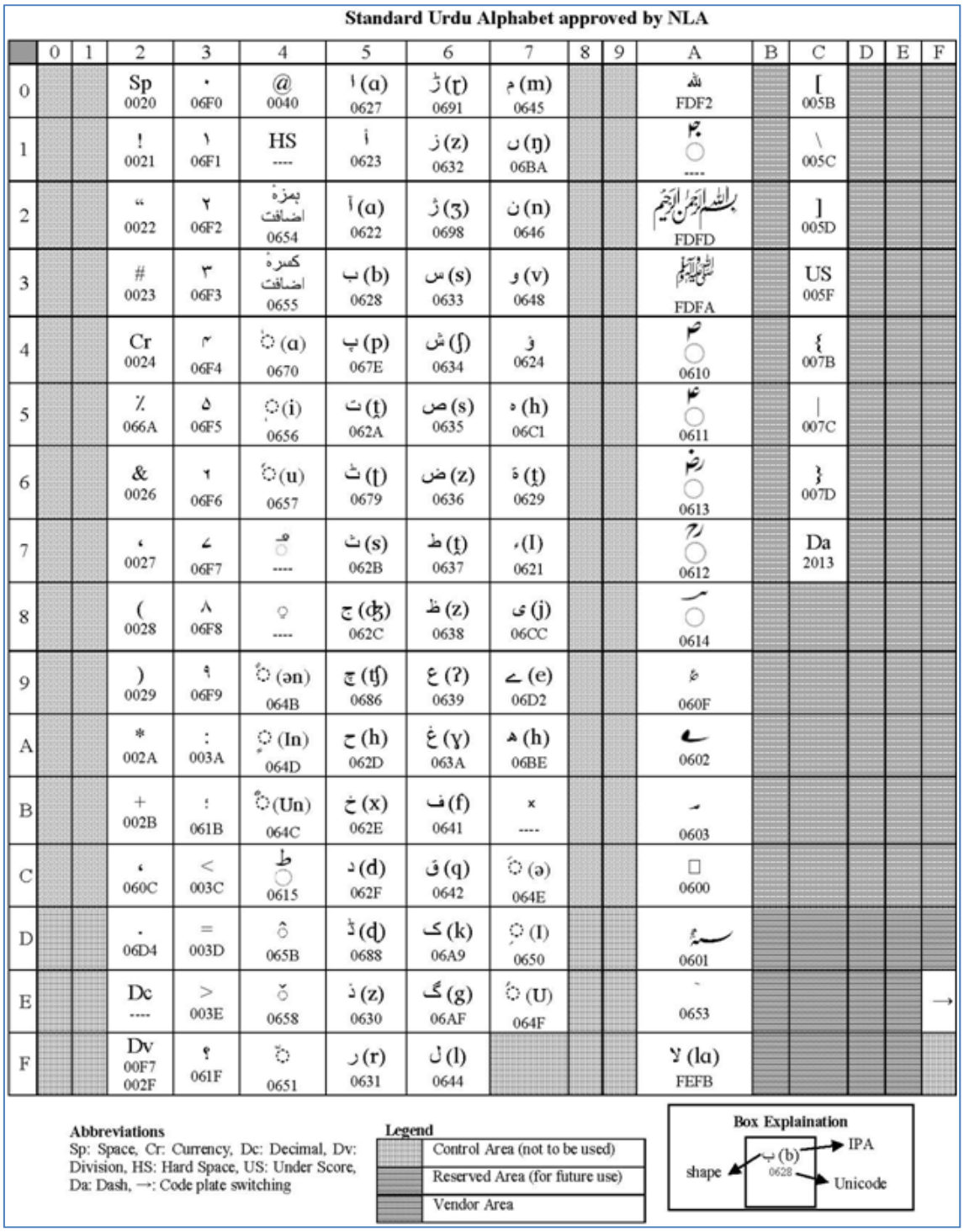


Figure 71: Urdu Zabta Takhti

This code page is kept similar to ASCII code (where possible). This is because people are familiar with the character distribution in ASCII as it is a worldwide standard. In addition, owing to its universal acceptability, many hardware and software systems (especially the earlier ones, some of which are still deployed) conform very closely to the ASCII standard [2].

Analysis of Urdu script

Consonants

There are two types of consonants in Urdu, non-aspirated consonants (simple consonants) and aspirated consonants. They are discussed separately below.

Non-aspirated consonants

There are 35 non-aspirated consonant letters representing 27 consonant sounds. In Urdu, one consonant sound may be represented by two or more characters, e.g. the sound [s] is represented by 3 different letters, Seh (ث), Seen (س) and Sad (ص). Here, "consonant" means symbol not sound. Out of 35 consonants, 32 consonants are adopted from the Persian alphabet. Retroflex sounds are not present in Persian, but they exist in Urdu and other languages of the Indian subcontinent. 3 retroflex consonants are added to the 32 consonants of Persian to accommodate the missing sounds. Shapes for these sounds were derived from the existing characters at some point back in history. These characters are Tteh (ٹ) [ʈ], Ddal (ڈ) [ɖ] and Rreh (ڑ) [ɽ]. In other categories, Urdu consonants contain velar, palatal, dental, labial, sibilant and glottal consonants. These consonants are listed in the table below with their corresponding Unicode values in hexadecimal and their IPAs.

Sr.	Character	Unicode	Sr.	Character	Unicode	Sr.	Character	Unicode
1	ب [b]	0628	13	ر [r]	0631	25	ف [f]	0641
2	پ [p]	067E	14	ڑ [ɽ]	0691	26	ق [q]	0642
3	ت [t̪]	062A	15	ز [z]	0632	27	ک [k]	06A9
4	ٹ [ʈ]	0679	16	ژ [z]	0698	28	گ [g]	06AF
5	ث [s]	062B	17	س [s]	0633	29	ل [l]	0644
6	ج [ʤ]	062C	18	ش [ʃ]	0634	30	م [m]	0645
7	چ [ʧ]	0686	19	ص [s]	0635	31	ن [n]	0646
8	ح [h]	062D	20	ض [z]	0636	32	و [v]	0648
9	خ [x]	062E	21	ط [t̪]	0637	33	ہ [h]	06C1
10	د [d̪]	062F	22	ظ [z]	0638	34	ی [j]	06CC
11	ڈ [ɖ]	0688	23	ع [ʔ]	0639	35	ۃ [t̪]	0629
12	ذ [z]	0630	24	غ [ɣ]	063A			

Table 62: Non-aspirated Urdu consonants

Aspirate consonants

The phenomenon of aspiration also does not exist in Persian. On the other hand, aspiration exixts in languages of Indian subcontinent, like Punjabi, Urdu, Hindi, etc. In Urdu, there exist 15 aspirated consonants. Unlike retroflex sounds, aspirates are not assigned separate characters. A special letter called Heh Doachashmee (ھ) is used to mark the aspiration. Aspirates are represented as the combination of a consonant to be aspirated and Heh Doachashmee (ھ) e.g. ب [b] + ھ [h] = بھ [bʰ], ج [ʤ] + ھ [h] = جھ [ʤʰ], etc. All 15 aspirates of Urdu are listed in Table 63.

Sr.	Character	Sr.	Character	Sr.	Character	Sr.	Character
1	بھ [bʰ]	5	جھ [ʤʰ]	9	کھ [kʰ]	13	مھ [mʰ]
2	پھ [pʰ]	6	چھ [ʧʰ]	10	گھ [gʰ]	14	لھ [lʰ]
3	تھ [ʈʰ]	7	دھ [ɖʰ]	11	رھ [rʰ]	15	نھ [nʰ]
4	ٹھ [ʈʰ]	8	ڈھ [ɖʰ]	12	ڑھ [ɽʰ]		

Table 63: Aspirated Urdu consonants

The first 10 aspirates of the table above are present in a very large number of words as compared to the remaining 5 aspirates. There is no doubt about the existence of first set of 10 aspirates. But there are doubts about the existence of others. Khan [95] has pointed out the existence of aspirates [rʰ], [ɽʰ], [lʰ], [mʰ] and [nʰ] in Urdu. He has mentioned some words which are believed to have these aspirates. According to him, the words سرھانہ [sərʰanɑ] (pillow), بارھواں [bɑrʰəvɑɲ] (twelfth) contain an aspirated [rʰ]. He also pointed out that [ɽʰ] is aspirated in the following words ساڑھی [sɑɽʰi] (an indian dress for ladies), بوڑھا [buɽʰɑ] (old man). Likewise he has given words کولھو [kolʰu] (oil expeller), کلھاڑی [kʊlʰɑɽi] (axe) for aspirated [lʰ]. For aspirated [mʰ], he has given the word کمھار [kʊmʰɑr] (potter) and for aspirated [nʰ], he has given words ننھا [nənʰɑ] (tiny) and کنھیا [kənʰəjɑ] (name of Hindu god Krishna, means beautiful boy).

As a small exercise, we looked up these words in two dictionaries (1) "*A Dictionary of Urdu, Classical Hindi and English*" [151] and (2) "فیروز اللغات" (An Urdu Dictionary) [48]. All above mentioned words and with the same spellings are present in the first dictionary (an old dictionary, compiled in 1911) except the word بارھواں [bɑrʰəvɑɳ]. On the other hand, only four words with the same spellings are present in the second dictionary (the new one). In remaining words, the aspiration marker Heh Doachashmee (ھ) is replaced by Heh [h] (ہ). The word بارھواں [bɑrʰəvɑɳ] is not found at all in either dictionaries. Both dictionaries include words containing aspirates [rʰ], [ɽʰ], [lʰ], [mʰ] and [nʰ]. Therefore, we cannot rule out the existence of these aspirates in Urdu.

Vowels

Urdu has 10 vowels. Seven of these 10 vowels also have nasalized forms [78] and this raises the total number of vowels to 17. Unlike English and French, Urdu has not assigned separate characters to vowels. In Urdu, vowels are represented with the help of four long vowel characters Choti Yeh (ی), Vav (و), Alef (ا) and Alef Madda (آ) and with the help of diacritical marks (Arabic Fatha – Zabar ◌َ , Arabic Damma – Pesh ◌ُ and Arabic Kasra – Zer ◌ِ). The representation of a vowel is context-sensitive, i.e. a vowel may be represented in two or more ways according to the context in a word. Nasalization of a vowel is marked with Noon Ghunna (ں). Noon Ghunna always comes at the end of a word. In the middle of a word, nasalization is done by Noon (ن). To distinguish between Noon and nasalized Noon, a diacritical mark (◌ٜ) is placed above the dot of a nasalized Noon, e.g. آنسو [ɑɲsu] (tears), لونڈ [lɔɲɖ] (excessive), etc. This Noon Ghunna diacritical mark (◌ٜ) is normally dropped by common people and is not present in common texts. Urdu vowels and their analysis have previously been given in Table 11.

Diacritical Marks

Urdu is very rich in diacritical marks, the backbone of its vowel system. There are 15 diacritical marks in Urdu. Zer (◌ِ), Zabar (◌َ), Pesh (◌ُ), etc. play an important role in the construction of Urdu vowel system as previously discussed. Mainly, diacritical marks represent the vowel sounds, but they are also used for other purposes. For example, one of these diacritical marks (◌ّ) is used to geminate the sound of a consonant, and two diacritics (◌ِ, ◌ُ) are used to build compound words. Jazam (◌ْ) is used to mark the absence of a vowel after the base consonant.

Zer, Zabar and Pesh when doubled are pronounced with added of the sound [n]. For example, Do-Zabar (◌ً), Do-Zer (◌ٍ) and Do-Pesh (◌ٌ) represent sounds [ən], [ɪn] and [ʊn], respectively,

where ‘Do’ means two. This is called ‘Tanwin’ and takes place only at the end of Arabic loan words [152]. These “Tanwins” rarely occur in Urdu [152].

Analysis of diacritical marks is given in Table 64.

Sr.	Diacritic	Analysis
1	ZABAR ◌َ	Zabar represents the vowel sound [ə] and also represents the vowels [ɔ] and [æ] in certain contexts, discussed in Table 11.
2	ZER ◌ِ	Zer represents the vowel sound [ɪ] and also represents the vowel [i] in a certain context.
3	PESH ◌ُ	Pesh represents the vowel sound [ʊ] and also represents the vowel [u].
4	SHADDA ◌ّ	In Arabic, Persian and Urdu, Shadda is used to geminate the sound of a consonant [152].
5	HAMZA-e-IZAFAT ◌ٔ	Hamza-e-Izafat is represented by a small Hamza mark above and is used to connect two words to form a compound word.
6	KASR-e-IZAFAT ◌ِ	Kasr-e-Izafat is represented by Zer (G.) at the end of a word and is used to connect two words to form a compound word.
7	JAZM ◌ْ	Jazm is used to mark the absence of a vowel after the base consonant (Platts 1909). It has same effect as ARABIC SUKUN (Unicode 0652).
8	NOON-GHUNNA MARK ◌ں	Already discussed above in this section.
9	KHARI ZABAR ◌ٰ	In some Arabic loan words, Khari Zabar is used in the middle or at the end of an Arabic loan word to represent the vowel [ɑ].
10	KHARI ZER ◌ٖ	Khari Zer represents the vowel sound [i]. It is rarely used in Urdu and is used in only a few Arabic loan words.
11	ULTA PESH ◌ٗ	Ulta Pesh represents the vowel sound [u]. It is also rarely used in Urdu in a few Arabic loan words.
12	CHOTA TOAY ◌ٙ	It is not used commonly. It is used in a few Arabic loan words only.
13	DO-ZABAR ◌ً	It is used with Alef (ا) and Gol Teh (ہ) at the end of a word to produce the sound [ən].
14	DP-PESH ◌ٌ	It represents the vowel sounds [ʊn]. It is used at the end of a few Arabic loan words only.
15	DO-ZER ◌ٍ	It represents the vowel sounds [ɪn]. It is used at the end of a few Arabic loan words only.

Table 64: Analysis of diacritical marks

HAMZA

HAMZA (ء) is a placeholder and is used to separate two successive vowel sounds. In the example word آسائش [ɑsɑɪʃ] (comfort), HAMZA is separating two vowel sounds [ɑ] (Alef) and [ɪ] (Zer) and in the word آؤ [ɑo] (come), HAMZA is separating two vowel sounds [ɑ] (Alef Madda) and [o] (Vav) In the example word آسائش [ɑsɑɪʃ] (comfort), HAMZA is separating the two vowel sounds Alef and Zer, but normally common people drop Zer and write آسائش instead of آسائش and HAMZA plays the role of Zer. Hence HAMZA may be considered as a vowel.

Analysis of Punjabi/Shahmukhi

Punjabi/Shahmukhi alphabet is a superset of the Urdu alphabet. It has one additional non-aspirated consonant, the retroflex Noon, Rnoon (ڻ) [ɳ] [117, 119]. The rest is the same as Urdu. Punjabi is also traditionally written in Nasta’leeq style.

Analysis of Seraiki/Shahmukhi

Seraiki/Shahmukhi is again a superset of Punjabi. It has three additional characters. The rest is the same as Urdu and Punjabi/Shahmukhi. It is traditionally written in Nasta'leeq style.

Analysis of Kashmiri/Urdu

Kashmiri employs the Urdu alphabet with a few additions to represent its specific vowels. Kashmiri has two additional Yehs (ى), one with an oval below (ۍ) and the other with a 'v' mark above (ێ). It also has two additional Waws (و), one with a circle at the ending tail (ۄ) and the other with a 'v' mark above (ۅ). It has two more diacritical marks, a slightly modified Hamza (ء) written above and below the character. The extra characters of Kashmiri are shown in Table 65. It is also traditionally written in Nasta'leeq style.

Sr.	Symbol	Unicode	Sr.	Symbol	Unicode
1	ۍ []	-	4	ێ [e]	06CE
2	ۄ [ɔ]	06C4	5	ٗ [ə]	-
3	ۅ [o:]	06C6		ٕ []	-

Table 65: Kashmiri characters

Analysis of Sindhi/Sindhi

Sindhi has 40 non-aspirated consonants and 11 aspirated consonants. In the Sindhi script, aspiration is expressed in different ways. For example, the aspiration of Jeem (ج) is indicated by Heh Doachashmee (ھ) like in Urdu and Punjabi, and the aspiration of Beh (ب) is expressed by a separate new character with four dots below پ. Sindhi aspirated and non-aspirated consonants that are not present in Urdu or have different shapes from those in Urdu are given in Table 66.

Sr.	Symbol	Unicode	Sr.	Symbol	Unicode
1	ٻ [ɓ]	067B	12	ڈھ [ɖʰ]	068D
2	پھ [bʰ]	0680	13	ڙ [ɽ]	0699
3	ٽھ [ʈʰ]	067F	14	ڑھ [ɽʰ]	-
4	ٽ [ʈ]	067D	15	ڦ [pʰ]	06A6
5	ٹھ [ʈʰ]	067A	16	ک [k]	06AA
6	ڇ []	0684	17	ک [kʰ]	06A9
7	ڃ [ɲ]	0683	18	ڳ [ɠ]	06B3
8	ڇھ [ʧʰ]	0687	19	گ [ŋ]	06B1
9	ڈھ [ɖʰ]	068C	20	ڻ [ɳ]	06BB
10	ڈ [ɖ]	068A	21	ي [j]	064A
11	ڈ [ɗ]	068F			

Table 66: Aspirated and non-aspirated Sindhi consonants

Sindhi has 16 vowels that are also context-sensitive. Sindhi is traditionally written in Naskh.

Annex 2 Analysis of the Hindi Writing Systems

Hindi, an Indo-European language, enjoys the status of being the mother tongue of 366 million persons of India [156]. It is one of the official languages of India and is used as the language of administration, media, education and literature in Delhi, Uttar Pradesh, Bihar, Madhya Pradesh, Rajasthan, Haryana and Himachal Pradesh. Elsewhere in India, Hindi is used, alongside English, as a second language. Hindi is also spoken in Bangladesh, Belize, Botswana, Canada, Germany, Guyana, Kenya, Nepal, New Zealand, the Philippines, Singapore, South Africa, Surinam, Trinidad, Uganda, UAE, UK, USA, Yemen, and Zambia. The total count of persons who can speak and understand Hindi is more than 850 millions [156].

Hindi Alphabet

Historical emergence of the Hindi/Devanagari script

The Hindi alphabet known as Devanagri is a simplified version of the alphabet most used for Sanskrit, Nagari (literally "urban") or Devanagari ("godly urban") which evolved from the Brahmi writing system used in Ashoka's times (3rd century BC). During the six following centuries, Brahmi evolved into two distinct subtypes, the northern and the southern ones. Between the 6th and the 10th century, the northern subtype in the form of the Gupta script, used during the Gupta dynasty (4th and 5th century), evolved into a central subtype known as the Kutila ("bent") or Kutiya, also called Sidhamatrka ("with complete vowels"), a cursive version of the Gupta script. Kutila evolved into early Nagari and proto-Bengali (used for Maithili and modern Bengali) scripts. Early Nagari developed into the modern Nagari (used for Hindi, Marathi, Nepali), Kaithi used by the Kayasth cast of writers and clerks, Gujarati (19th century) and Modi, a cursive type used in Shivaji's times (18th century) for writing Marathi. Parallel to this northern and central subtype, the western subtype evolved into Sharda, Landa (used by the merchant and clerical Hindu communities in Panjab and Sindh) and Takri (used in Himachal Pradesh and Jammu) scripts. Gurmukhi, a script used for writing Punjabi in India, is a derivation of Sharda, Landa and Takri. Nagari imposed itself as the main regional and then the national (India's constitution 1950) script for Hindi [136].

Features of Devanagari script

Hindi/Devanagari is read and written left-to-right. All graphic systems derived from the Brahmi have been termed syllabic alphabets. Compared with other syllabic scripts like the Japanese hiragana and katakana, which have one symbol for [kə] and different symbols for [kɪ] and [kʊ], it is clear that Hindi/Devanagari script is partly syllabic as the graphic symbol for a consonant only inherits the vowel sound [ə]. [94, 136] All other vowels are noted as consonant + vowel diacritical mark e.g. क [k] + ि [ɪ] = कि [kɪ], क [k] + ु [ʊ] = कु [kʊ], etc.

Consonants do not have the upper and lower cases and are always pronounced exactly the same way unlike English. There are 40 consonants in Hindi. All consonants inherit the vowel sound [ə] and the inherent [ə] is silent after the final consonant [94, 136].

All vowels have two forms except the short vowel [ə], one being the diacritical mark and the other being the independent character. The diacritical form of a vowel is used when a vowel appears in the middle or at the end of a word or a syllable. Diacritical marks are written before, after, under or above a consonant. The independent character form of a vowel is used when it occurs at the start of a word or a syllable. The independent form is written as a separate character, e.g. in आब [ɑb] (water), the vowel sound [ɑ] is represented by the character आ [94, 136].

The feature that makes the Devanagari much more complex is that two or more consonants can be combined together to form a consonant cluster. The cluster form is used to denote the non-intervention of the inherent [ə] or another vowel between two or more consonants e.g. हिन्दी [hɪndi] (Hindi), ग्वाला [gvɑlɑ] (cowherd) (consonant clusters are underlined), etc. The cluster form is normally known as 'Conjunct'. A example Hindi sentence in Devanagari is shown below:

हिन्दी हिन्दूस्तान की सरकारी ज़ुबान है

Hindi is the official language of India

Figure 72: An example Hindi sentence

Standard codepage of Hindi

The Hindi/Devanagari script has 40 consonants, 11 independent vowel characters, 10 vowel diacritical marks, 10 digits, etc. It is read and written from left to right. In Unicode standard 4.1, The Hindi alphabet is assigned 128 code points, 0900 – 097F. The Hindi code page is given in Table 67.

	090	**091**	**092**	**093**	**094**	**095**	**096**	**097**
0		ऐ 0910	ठ 0920	र 0930	ी 0940	ॐ 0950	ॠ 0960	॰ 0970
1	ँ 0901	ऑ 0911	ड 0921	ऱ 0931	ु 0941	॑ 0951	ॡ 0961	
2	ं 0902	ओ 0912	ढ 0922	ल 0932	ू 0942	॒ 0952	ॢ 0962	
3	ः 0903	ओ 0913	ण 0923	ळ 0933	ृ 0943	॓ 0953	ॣ 0963	
4	अ 0904	औ 0914	त 0924	ऴ 0934	ॄ 0944	॔ 0954	। 0964	
5	अ 0905	क 0915	थ 0925	व 0935	ॅ 0945		॥ 0965	
6	आ 0906	ख 0916	द 0926	श 0936	ॆ 0946		० 0966	
7	इ 0907	ग 0917	ध 0927	ष 0937	े 0947		१ 0967	
8	ई 0908	घ 0918	न 0928	स 0938	ै 0948	क़ 0958	२ 0968	
9	उ 0909	ङ 0919	ऩ 0929	ह 0939	ॉ 0949	ख़ 0959	३ 0969	
A	ऊ 090A	च 091A	प 092A		ॊ 094A	ग़ 095A	४ 096A	
B	ऋ 090B	छ 091B	फ 092B		ो 094B	ज़ 095B	५ 096B	
C	ऌ 090C	ज 091C	ब 092C	् 093C	ौ 094C	ड़ 095C	६ 096C	
D	ऍ 090D	झ 091D	भ 092D	ऽ 093D	् 094D	ढ़ 095D	७ 096D	□ 097D
E	ऎ 090E	ञ 091E	म 092E	ा 093E		फ़ 095E	८ 096E	
F	ए 090F	ट 091F	य 092F	ि 093F		य़ 095F	९ 096F	

Table 67: Standard codepage of Hindi (Unicode)

Analysis of Devanagari for Hindi

This section contains a detailed analysis of the Hindi/Devanagari writing system.

Vowels

All Hindi vowels can be represented in two different ways except the vowel [ə], one being the diacritical form and other being the independent character form. Hindi contains 11 main vowels. According to Montaut [136], all vowels can be nasalized except the vowel [r̥]. Hence the total number of vowels is 21. On the basis of ways of vowel representation, Hindi vowels can be further subdivided into vowel diacritical marks and independent vowels. They are discussed and analyzed below.

Vowel diacritical marks

The diacritical form of a vowel is used when a vowel appears in the middle or at the end of a word or a syllable. Vowel diacritical marks are written before, after, under or above a consonant e.g. किताब [kɪt̪ab] (book), ज़मीन [zəmin] (earth), घूमान [gʰuməna] (to revolve around), मेरा [merɑ] (mine), आख़री [ɑxəri] (last), etc. The vowel diacritical mark for the vowel [ə] does not exist as each constant symbol inherits it [94, 136].

There are 10 vowel diacritical marks in Hindi (ा [ɑ], ि [ɪ], ी [i], ु [ʊ], ू [u], ृ [r̥], े [e], ै [æ], ो [o], ौ [ɔ]) and two symbols (ँ, ं) for nasalization. According to Kellogg [94], the nasalization mark 'Anunasik' (ँ) is used for the nasalization of a preceding vowel e.g. भसैं [bæ̃s] (buffalo), कहीं [kəhĩ] (somewhere), गाँ [gɑ̃] (cow), क्यों [kjõ] (why), etc. But Montaut [136] adds that 'Anunasik' (ँ) is replaced by a simple dot above 'Anusavar' (ं) when the vowel glyph goes over the upper line e.g. भसैं [bæ̃s] (buffalo), कहींं[kəhĩ] (somewhere), क्यों [kjõ] (why), etc. 'Anusavar' is a much stronger nasalized form than 'Anunasik'. 'Anunasik' and 'Anusavar' both denote the nasalization of a preceding vowel, and therefore never begin a syllable or a word. The diacritical mark ॄ [r̥] is used only in Sanskrit loan words and is very rare in Hindi texts. Vowel diacritical marks and their analysis with examples are given in Table 68.

Sr.	Vowel Diacritical Mark	Analysis
1	Diacritical mark of the vowel [ə] does not exist.	All consonants inherit the vowel [ə] hence it does not have a vowel mark e.g. करण [kərəɳ] (ear), रब्ब [rəbb] (God)
2	ा [ɑ]	It comes after a consonant e.g. गाना [gɑnɑ] (to sing), जाना [ʤɑnɑ] (to go)
3	ि [ɪ]	It comes before a consonant e.g. दिल्ल [d̪ɪll] (heart), किरण [kɪrəɳ] (ray)
4	ी [i]	It comes after a consonant e.g. अमीर [əmir] (rich), क़रीब [qərib] (near)
5	ु [ʊ]	It comes below a consonant e.g. सुन्दर [sʊnd̪ər] (beautiful), सुनना [sʊnnɑ] (to hear)
6	ू [u]	It comes below a consonant e.g. करतूत [kərt̪ut̪] (deed), सूरत [surət̪] (face)
7	ृ [r̥]	It comes below a consonant e.g. कृपाण [kr̥pɑɳ] (sword), कृष्ण [kr̥ʃn] (the god Krishna of Hindus)
8	े [e]	It comes above a consonant e.g. मेरा [merɑ] (mine), तेरा [t̪erɑ] (your)
9	ै [æ]	It comes above a consonant e.g. फैलाना [pʰælɑnɑ] (to stretch), पैदल [pæd̪əl] (on foot)
10	ो [o]	It comes after a consonant e.g. रोना [ronɑ] (to weep), रोग [rog] (malady)
11	ौ [ɔ]	It comes after a consonant e.g. मौत [mɔt̪] (death), रौंदना [rɔ̃d̪nɑ] (to trample)
12	C + ं [ə̃] (C represents a consonant)	The vowel [ə] is inherited by all consonants thus nasalized form of the vowel [ə] is represented by 'Anunasik' (ँ) or 'Anusavar' (ं), placed above a consonant (just a dot above a consonant) e.g. नंगा or नँगा [nə̃gɑ] (naked), रंगीन or रँगीन [rə̃gin] (colorful)
13	ा + ँ or ं [ɑ̃]	गाँ or गां [gɑ̃] (cow), साँग or सांग [sɑ̃g] (disguise)
14	ि + ँ or ं [ɪ̃]	
15	ी + ँ or ं [ĩ]	कहीं or कहीं [kəhĩ] (somewhere), नहीं or नहीं [nəhĩ] (no)
16	ु + ँ or ं [ʊ̃]	मुँजा [mʊ̃ʤɑ] (a thread like worm)
17	ू + ँ or ं [ũ]	मूँछ or मूंछ [mũʧʰ] (moustache)
18	े + ँ or ं [ẽ]	में or मैं [mẽ] (in), रहें or रहें [rəhẽ] (live).
19	ै + ँ or ं [æ̃]	भैंस or भैंस [bæ̃s] (buffalo), मैं or मैं [mæ̃] (I)
20	ो + ँ or ं [õ]	सोंधा or सोंधा [sõd̪ʰɑ] (fragrant), दोंकना or दोंकना [d̪õkənɑ] (to roar)
21	ौ + ँ or ं [ɔ̃]	रौंदना or रौंदना [rɔ̃d̪nɑ] (to trample), बौंगा or बौंगा [bɔ̃gɑ] (silly boy)

Table 68: Analysis of devangari vowel dicritics

Independent Vowels

The independent vowel character is used when a vowel appears at the start of a word or a syllable. All 11 main vowels have their independent character forms including the vowel [ə]. Their nasalized versions are derived in the same way as they are derived for vowel diacritical marks in the previous section. The independent vowel character is used when it comes at the start of a word e.g. ईमान [imɑn] (faith), अग्गर [əggər] (if), एक [ek] (one), etc. or when it comes at the start of a syllable e.g. दाइरा [d̪ɑɪrɑ] (circle) contains two syllables [d̪ɑ] and [ɪrɑ] and the vowel [ɪ] at the beginning of second syllable is written by the independent form इ, नज़रअंदाज़ [nəzərə̃d̪ɑz] (ignore) contains syllables [nə], [zə], [rə], and [ə̃dɑz] and at the beginning of the fourth syllable the vowel [ə̃] is written by the independent vowel form अं , etc. It is clear from examples above that the independent vowel form is used at the beginning of a syllable when it begins with a vowel and its preceding syllable ends in a vowel. In other words, the independent vowel character is used for a vowel when two vowels appear consecutively in the middle of a word.

In Arabic loan words, the independent vowel character is used to transcribe Ain (ع), e.g. तबीअत [ʈəbijʔʈ] (physical status) = طبعیت, तअमीर [ʈʔmir] (build) تعمیر, etc. This creates confusions that an independent vowel character actually represents a vowel sound or Ain (ع).

Hindi vowels with examples are given in Table 69.

Sr.	Vowel Character	Examples
1	अ [ə]	अच्छा [əʧʧʰɑ] (good), नज़रअंदाज़ [nəzərə̃dɑz] (ignore), etc.
2	आ [ɑ]	आग [ɑg] (fire), ख़ूदआराई [xʊɖɑrɑi] (proud), etc.
3	इ [ɪ]	इकलौता [ɪklɔʈɑ] (only son), दाइरा [ɖɑɪrɑ] (circle), etc.
4	ई [i]	ईमान [imɑn] (faith), किठनाईयों [kəʈɪnɑijõ] (hard lucks), etc.
5	उ [ʊ]	उद्धर [ʊɖɖʰər] (there), नोउमर [noʊmər] (newly young), etc.
6	ऊ [u]	ऊधम [uɖʰəm] (noise), उठाऊं [ʊʈɑũ] (asking permission to lift), etc.
7	ॠ [ɾ̥]	ॠया [prɾ̥jɑ] (decision committee)
8	ए [e]	एक [ek] (one), पाए [pɑe] (may get), etc.
9	ऐ [æ]	ऐसा [æsɑ] (like this)
10	ओ [o]	ओझल [oʤʰəl] (out og sight), िदशाओं [dɪʃɑõ] (directions), etc.
11	औ [ɔ]	औसत [ɔsəʈ] (average), औटाना [ɔʈɑnɑ] (to boil), etc.
12	अ [ə̃]	अँगारा [ə̃gɑrɑ] (spark), अँमेज़ [ə̃grez] (Englishman), etc.
13	आ [ɑ̃]	आँख [ɑ̃kʰ] (eye), आँगन [ɑ̃gən] (courtyard), etc
14	इ [ĩ]	इँच [ĩʧ] (inch), इँचना [ĩʧɔnɑ] (to be attracted), etc.
15	ई [ĩ]	ईन्धन [ĩnɖʰən] (fuel), ईंट [ĩʈ] (brick), etc.
16	उ [ʊ̃]	उँगली [ʊ̃gli] (finger), उँ ट [ʊ̃ʈ] (camal), etc.
17	ऊ [ũ]	ऊँ चा [ũʧɑ] (High), ऊँ घ [ũgʰ] (dozing), etc.
18	ए [ẽ]	एँचा [ẽʧɑ] (pulled)
19	ऐ [æ̃]	ऐँठना [æ̃ʈʰənɑ] (to twist), ऐँडा [æ̃ɖɑ] (rolling), etc.
20	ओ [õ]	ओँडाई [õɖɑi] (depth)
21	औ [ɔ̃]	औँधा [ɔ̃ɖʰɑ] (with the face down), औँगी [ɔ̃gi] (silence), etc.

Table 69:Analysis of Devangari vowel characters

Consonants

Hindi consonants can be divided into non-aspirated consonants and aspirated consonants. The other main thing relative to consonants is the conjunct form, a cluster of two or more consonants. These are discussed separately below.

Non-aspirated consonants

There are 29 non-aspirated consonant characters that represent 28 consonant sounds. The sound [ʃ] is represented by two characters श and ष. Hindi/Devanagari consonants contain velar, palatal, dental, labial, sibilant, retroflex and glottal consonants like Urdu/Urdu. These consonants are given in Table 70.

Sr.	Character	Unicode	Sr.	Character	Unicode	Sr.	Character	Unicode
1	क [k]	0915	11	द [d̪]	0926	21	ष [ʃ]	0937
2	ग [g]	0917	12	न [n]	0928	22	स [s]	0938
3	ङ [ŋg]	0919	13	प [p]	092A	23	ह [h]	0939
4	च [ʧ]	091A	14	ब [b]	092C	24	क़ [q]	0958
5	ज [ʤ]	091C	15	म [m]	092E	25	ख़ [x]	0959
6	ञ [ŋʤ]	091E	16	य [j]	092F	26	ग़ [ɣ]	095A
7	ट [ʈ]	091F	17	र [r]	0930	27	ज़ [z]	095B
8	ड [ɖ]	0921	18	ल [l]	0932	28	ड़ [ɽ]	095C
9	ण [ɳ]	0923	19	व [v]	0935	29	फ़ [f]	095E
10	त [t̪]	0924	20	श [ʃ]	0936			

Table 70: Non-aspirated Hindi consonants

Aspirated consonants

There are a total of 15 aspirates in Hindi. They are represented in two different ways. Unlike Urdu, some of them are represented by separate characters. There are 11 such aspirated consonants in Hindi that are represented by separate characters. They are listed in Table 71.

Sr.	Character	Unicode	Sr.	Character	Unicode	Sr.	Character	Unicode
1	ख [kʰ]	0916	5	ठ [ʈʰ]	0920	9	फ [pʰ]	092B
2	घ [gʰ]	0918	6	ढ [ɖʰ]	0922	10	भ [bʰ]	092D
3	छ [ʧʰ]	091B	7	थ [t̪ʰ]	0925	11	ढ़ [ɽʰ]	095D
4	झ [ʤʰ]	091D	8	ध [d̪ʰ]	0927			

Table 71: Aspirated Hindi consonants

Remaining consonants are represented by the combination of a consonant to be aspirated and the conjunct form of Ha (ह) e.g. दल्हन[dʊlʰn] (bride), कुम्हार [kʊmʰɑr] (potter), नन्हा [d̪ʊlhən] (bride)). They are treated by some linguists as similar to other aspirated consonants ([kʊmʰɑr], [nənʰɑ], [d̪ʊlʰən]) and not to the sequence CC (C + h)" (where C stands for consonant).

Conjunct forms

Conjunct forms of consonants are used when no intervening vowel separates two or more consonants [94, 136]. This makes Devanagari much more complex as both consonants in contact may modify their form. In writing, clusters are usually formed by the unmodified form of the second consonant and the shortened form of the first. There are 6 different ways to form clusters in Hindi:

1. When the first contains a final vertical bar, this bar is dropped, e.g. the cluster of ग [g] and ल [l] is ग्ल [gl] and it is formed by omitting the final vertical bar of ग, the cluster of त [t̪] and ल [l] is त्ल [t̪l] and it is formed by omitting the final vertical bar of त, etc.
2. When the first letter has a round shape, both consonants are written vertically with their entire graph e.g. ट्ट, ड्ड, etc.
3. When the first letter is ह [h], the second starts in the low curve e.g. the cluster of ह [h] and य [j] is ह्य.

4. Cluster forms of र [r] are the most diverse. When र [r] is the first consonant of a cluster, the sign - is placed above the second consonant, e.g. the cluster of र [r] and क [k] is क्र [rk], the cluster of र [r] and व [v] is व्र [rv], etc. When र [r] is the second consonant of a cluster, then with some consonants, the sign ` is used e.g. the cluster of क [k] and र [r] is ब [kr], the cluster of घ [g^h] and र [r] is य [g^hr], etc. With some consonants, the sign ः is used, e.g. the cluster of ढ [$ɖ^h$] and र [r] is ह [$ɖ^h$r], the cluster of ड [ɖ] and र [r] is स [ɖr], etc.
5. The consonant त [t̪] has special forms of conjuncts with र [r], क [k] and itself. When र [r] is clustered with त [t̪], it also changes the shape of त [t̪] and the cluster form is ऽ [t̪r]. Conjunct forms of त [t̪] with क [k] and itself are क्त [kt̪] and त्त [tt̪] respectively.
6. The cluster form क्ष [kʃ] (क [k] + ष [ʃ] = क्ष [kʃ]) occurs only in Sanskrit loan words e.g. अक्षर [əkʃr] (letter), क्षमा [kʃəmɑ] (pardon), etc.

Cluster forms may occur at the start, in the middle and at the end of word. The consonant ढ़ [$ɖ^h$] never forms a cluster. Computationally, cluster forms make Devanagari a quite complex writing system.

Annex 3 Analysis of Punjabi/Gurmukhi

Gurmukhi derives its character set from Landa (old script of Indian sub-continent) and was standardized by Guru Angad Dev (second Sikh Guru, 1504-1552) in the 16th century, when it contained 35 consonants. The word Gurmukhi literally means "from the mouth of Guru". The whole of the Guru Granth Sahib (Holy book of Sikhs) is written in Gurmukhi. Its alphabets are Abugida, as each consonant has an inherent vowel (a) that can be changed using vowel signs. It is a left to right script and unlike Shahmukhi, its characters do not assume different shapes and also do not have small and capital forms. An example sentence is give below:

ਪੰਜਾਬੀ ਮੇਰੀ ਮਾਣ ਜੋਗੀ ਮਾਂ ਬੋਲੀ ਏ

Figure 73: Example sentence of Punjabi in Gurmukhi script

Modern Gurmukhi has 41 consonants, 9 vowels symbols, 2 symbols for nasal sounds, 1 symbol that duplicates the sound of any consonant, 3 subjoined forms of the consonants Ra, Ha and Va and 1 post-base form of Ya. In Unicode, Gurmukhi sub-range is from U+0A00 to U+0A7F. This provides 128 code points for the Gurmukhi characters, of which only 77 are currently used (Unicode 4.0.1). In addition, Danda and Double Danda are contained in the Devanagari sub-range at U+0964 and U+0965 respectively.

The analysis of Punjabi/Gurmukhi vowels and consonants is the same as previously discussed for Hindi/Devanagari. Gurmukhi employs different character shapes, except for a few characters that are the same as in Devanagari.

Gurmukhi does not have conjunct forms like Hindi/Devanagari. It has only a very limited conjunct forms that are called ‘PAIREENs’ (Sub-joins). There are three PAIREEN in *Gurmukhi*, “HAAHAA”, “VAAVAA” and “RAARAA” shown in Table 5. In the case of PAIREEN VAAVAA and PAIREEN RAARAA, these sub-joins are used in those words that have been derived from Sanskrit and are very rarely used.

PAIREEN HAAHAA is more frequently used in writings that the other two PAIREENs. It is used to mark the aspiration of some characters as a conjunct form of HA is used to mark aspiration of some Hindi consonants, discussed previously.

Table 72 shows the standard Unicode codepage for Gurmukhi.

	0A0	0A1	0A2	0A3	0A4	0A5	0A6	0A7
0		ਅ	ਠ	ਰ	ੀ			ਂ
1	ਁ		ਡ		ੁ			ਃ
2	ਂ		ਢ	ਲ	ੂ			ਦ
3	ਃ	ਉ	ਣ	ਲ਼				ਉ
4		ਊ	ਤ					ੴ
5	ਅ	ਕ	ਥ	ਵ				
6	ਆ	ਖ	ਦ	ਸ਼			੦	
7	ਇ	ਗ	ਧ		ੇ		੧	
8	ਈ	ਘ	ਨ	ਸ	ੈ		੨	
9	ਉ	ਙ		ਹ		ਖ਼	੩	
A	ਊ	ਚ	ਪ			ਗ਼	੪	
B		ਛ	ਫ		ੋ	ਜ਼	੫	
C		ਜ	ਬ	੍	ੌ	ੜ	੬	
D		ਝ	ਭ		ੑ		੭	
E		ਞ	ਮ	ਾ		ਫ਼	੮	
F	ਏ	ਟ	ਯ	ਿ			੯	

Table 72: Unicode codepage for Gurmukhi

Annex 4 Analysis of Devanagari for Sindhi, Seraiki and Kashmiri

Sindhi and Seraiki have some consonants that are not present in Hindi. In the recent past, Unicode has added some extra characters derived from the previously available characters for Sindhi and the same set of characters is also used in Seraiki. For example, a character for the sound [ɓ] is derived from the Hindi character ब [b] by putting a line below the character ब़. These additional Sindhi and Seraiki characters are shown in Table.

Sr.	Character	Unicode
1	ब़ [ɓ]	097F
2	ड़ [ɗ]	097E
3	ज़ []	097C
4	ग़ [ɠ]	097B

The rest of the analysis of Devanagari for Sindhi and Seraiki is the same as previously discussed for Hindi.

Annex 5 UIT Mappings for Indo-Pak Languages

Urdu UIT Mappings

Urdu/Urdu is a subset of the Punjabi/Shahmukhi alphabet, and it will be covered in the mappings for Punjabi/Shahmukhi. Thus, we have covered Urdu UIT mappings in Punjabi/Shahmukhi UIT mappings tables, given below.

Punjabi UIT Mappings

Consonant mappings

Sr.	Shahmukhi	IPA	Gurmukhi	UIT	Sr.	Shahmukhi	IPA	Gurmukhi	UIT
1	ب	b	ਬ	b	27	ک	k	ਕ	k
2	پ	p	ਪ	p	28	گ	g	ਗ	g
3	ت	t̪	ਤ	t_d	29	ل	l	ਲ	l
4	ٹ	ʈ	ਟ	t`	30	م	m	ਮ	m
5	ث	s	ਸ	s1	31	ن	n	ਨ	n
6	ج	ʤ	ਜ	d_Z	32	ڻ	ɳ	ਣ	n`
7	چ	ʧ	ਚ	t_S	33	و	v	ਵ	v
8	ح	h	ਹ	h1	34	ہ	h	ਹ	h
9	خ	x	ਖ਼	x	35	ۃ	t̪	ਤ	t_d2
10	د	d̪	ਦ	d_d	36	ی	j	ਯ	j
11	ڈ	ɖ	ਡ	d`	37	ھ+ب	bʰ	ਭ	b_h
12	ذ	z	ਜ਼	z1	38	ھ+پ	pʰ	ਫ	p_h
13	ر	r	ਰ	r	39	ھ+ت	t̪ʰ	ਥ	t_d_h
14	ڑ	ɽ	ੜ	r`	40	ھ+ٹ	ʈʰ	ਠ	t`_h
15	ز	z	ਜ਼	z	41	ھ+ج	ʤʰ	ਝ	d_Z_h
16	ژ	ʒ	ਜ਼	Z	42	ھ+چ	ʧʰ	ਛ	t_S_h
17	س	s	ਸ	s	43	ھ+د	d̪ʰ	ਧ	d_d_h
18	ش	ʃ	ਸ਼	S	44	ھ+ڈ	ɖʰ	ਢ	d`_h
19	ص	s	ਸ	s2	45	ھ+ر	rʰ	ਰ+◌੍+ਹ	r_h
20	ض	z	ਜ਼	z2	46	ھ+ڑ	ɽʰ	ੜ+◌੍+ਹ	r`_h

21	ٹ	ʈ	ੜ	t_d1	47	ھ+ک	k^h	ਖ	k_h
22	ڈ	z	ज़	z3	48	ھ+گ	g^h	ਘ	g_h
23	ع	ʔ	◌ਾ+ਅ	?	49	ھ+ل	l^h	ਲ+◌੍+ਹ	l_h
24	غ	ɣ	ਗ਼	G	50	ھ+م	m^h	ਮ+◌੍+ਹ	m_h
25	ف	f	ਫ਼	f	51	ھ+ن	n^h	ਨ+◌੍+ਹ	n_h
26	ق	q	ਕ+◌	q	52	ھ+و	v^h	ਵ+◌੍+ਹ	v_h

Table 73: Punjabi/Shahmukhi consonants mappings

Vowel mappings

Sr.	Shahmukhi	IPA	Gurmukhi	UIT	Sr.	Shahmukhi	IPA	Gurmukhi	UIT
1	آ	ɑ	ਆ	A	13	ِ+ع	ɪ	ਇ	?I1
2	◌َ+ا ،ا	ə	ਅ	@	14	ُ+ع	ʊ	ੁ	?U1
3	ِ+ا	ɪ	ਇ	I	15	ا+ع	ɑ	ਆ	?A1
4	ُ+ا	ʊ	ੁ	U	16	ع+ا	ə, e	ਅ, ਏ	@?
5	ے+ا ،ی+ا	e	ਏ	e, e3	17	ی+ع	e	ਏ	?e1
6	ے+َ+ا ،ی+َ+ا	æ	ਐ	{, {3	18	ی+ِ+ع	i	ੀ	?i1
7	ی+ِ+ا	i	ੀ	i	19	ی+َ+ع	æ	ਐ	?{1
8	و+ُ+ا	u	ੂ	u	20	و+ع	o	ਓ	?o1
9	و+ا	o	ਓ	o	21	و+َ+ع	ɔ	ਔ	?O1
10	و+َ+ا	ɔ	ਔ	O	22	و+ُ+ع	u	ੂ	?u1
11	ع	ə	ਅ	?	23	ے+ع	e	ਏ	?e4
12	َ+ع	ə	ਅ	?	24	ے+َ+ع	æ	ਐ	?{4

Table 74: Vowel mappings at the start of a word

Sr.	Shahmukhi	IPA	Gurmukhi	UIT	Sr.	Shahmukhi	IPA	Gurmukhi	UIT
1	آ ،ا	ɑ	◌ਾ, ਆ	A	19	ِ	i	◌ਿ	i2
2	ً	ə	-	@	20	ُ	u	◌ੁ	u2
3	ِ	ɪ	◌ਿ	I	21	ٗ	-		
4	ُ	ʊ	◌ੁ	U	22	ٰ	ɑ	◌ਾ	A3
5	ے ،ی	e	◌ੇ	e, e4	23	ع	ə	ਅ	?
6	ے+َ ،ی+ً	æ	◌ੈ	{, {4	24	َ+ع	ə	ਅ	?
7	ی+ِ	i	◌ੀ	i1	25	ِ+ع	ɪ	ਇ	?I1
8	و+ُ	u	◌ੂ	u1	26	ُ+ع	ʊ	ੁ	?U1
9	و	o	◌ੋ	o1	27	ا+ع	ɑ	ਆ	?A1
10	و+ً	ɔ	◌ੌ	O1	28	ع+ا	ɑ	ਆ	A1?
11	ی+ء	i, e	◌ੀ, ◌ੇ	I2i1, I2e1	29	ی+ع	e	ਏ	?e1
12	و+ء	o, U, u	◌ੋ, ◌ੁ, ◌ੂ	I2o1, I2U1, I2u1	30	ی+ِ+ع	i	ੀ	?i1

13	ے+ء	e	ਏ	I2e4	31	ی + ◌َ + ع	æ	ਐ	?{1
14	◌ੰ	ən	ਨ	@n1	32	و+ع	o	ਓ	?o1
15	◌ੱ	ʊn	◌ੁ+ਨ	U1n1	33	و+◌َ+ع	ɔ	ਔ	?O1
16	◌ੁ	ɪn	◌ਿ+ਨ	I1n1	34	و+◌ُ+ع	u	ਊ	?u1
17	◌ੌ	-	◌ੱ		35	ے+ع	e	ਏ	?e4
18	◌ੈ	-	-		36	ے+◌َ+ع	æ	ਐ	?{4

Table 75: Vowel mappings in the middle or at the end of a word

Sr.	Shahmukhi	IPA	Gurmukhi	UIT	Sr.	Shahmukhi	IPA	Gurmukhi	UIT
1	ا، آ	ɑ	◌ਾ	A1, A	6	ی، ے	e	◌ੇ	e1, e4
2	◌ِ	ɪ	◌ਿ	I1	7	ی+◌ٔ، ے+◌َ	æ	◌ੈ	{1, {4
3	ی+◌ِ	i	◌ੀ	i1	8	و	o	◌ੋ	o1
4	◌ُ	ʊ	◌ੁ	U1	9	و+◌ٔ	ɔ	◌ੌ	O1
5	و+◌ُ	u	◌ੂ	u1					

Table 76: Vowel sign mappings

Other symbols

Sr.	Shahmukhi	IPA	Gurmukhi	UIT	Sr.	Shahmukhi	IPA	Gurmukhi	UIT
1	ں، ن	ŋ	◌ਂ	~	15	م+و+◌َ+ا	ɔ + m	ੴ	Om
2	ں، ن	ŋ	◌ੰ		16	۔	-	. (as decimal)	+DECIMAL
3	◌ੈ	-	◌ੑ	.	17	۔	-	।	+SSTBD
4	۰	-	੦	+GURZERO +URZERO	18	۔	-	॥	+DSTBD
5	۱	-	੧	+GURONE +URONE	19	0	-	0	+ZERO
6	۲	-	੨	+GURTWO +URTWO	20	1	-	1	+ONE
7	۳	-	੩	+GURTHREE +URTHREE	21	2	-	2	+TWO
8	۴	-	੪	+GRUFOUR +RUFOUR	22	3	-	3	+THREE
9	۵	-	੫	+GURFIVE +URFIVE	23	4	-	4	+FOUR
10	۶	-	੬	+GURSIX +URSIX	24	5	-	5	+FIVE
11	۷	-	੭	+GURSEVEN +URSEVEN	25	6	-	6	+SIX
12	۸	-	੮	+GUREIGHT +UREIGHT	26	7	-	7	+SEVEN
13	۹	-	੯	+GURNINE +URNINE	27	8	-	8	+EIGHT
14	◌ّ	-	◌ੱ		28	9	-	9	+NINE

Table 77: Other symbol mappings

Seraiki UIT Mappings

Seraiki possesses four characters not in Pujabi/Shahmukhi. These additional characters and their UIT mappings are given in Table below.

Sr.	Seraiki/Devanagari	IPA	Seraiki/Shahmukhi	UIT
1	ब॒	ɓ	ٻ	b1
2	ड॒	ɗ	ڈ	d`1
3	ज़॒		ڄ	d_Z1
4	ग॒	ɠ	ڳ	g1

Table 78: Seraiki mappings for additional characters

Sindhi UIT Mappings

A large majority of characters of Sindhi have already been covered in UIT mappings for Punjabi. The additional character mappings are shown in the table below.

Sr.	Sindhi	IPA	Devanagari	UIT	Sr.	Sindhi	IPA	Devanagari	UIT
1	ٻ	ɓ	ब॒	b1	12	ڍ	ɖʱ	ढ	d`_h
2	ڀ	bʰ	भ	b_h	13	ڑ	ɽ	ड़	r`
3	ٽ	ʈ̪ʰ	थ	t_d_h	14	ڑھ	ɽʰ	ढ़	r`_h
4	ٹ	ʈ	ट	t`	15	ڦ	pʰ	फ	p_h
5	ٺ	ʈʰ	ठ	t`_h	16	ک	k	क	k
6	ڄ		ज़॒	d_Z1	17	ک	kʰ	ख	k_h
7	ڃ	ɲ	ञ	N2	18	ڳ	ɠ	ग॒	g1
8	ڇ	ʧʰ	छ	d_d	19	ڱ	ŋ	ङ	N1
9	ڌ	ɖ̪ʰ	ध	d_d_h	20	ڻ	ɳ	ण	n`
10	ڊ	ɖ	ड	d`	21	ي	j	य	j
11	ڙ	ɗ	ड॒	d`1					

Table 79: UIT mappings of additional Sindhi/Sindhi and Sindhi/Devanagari characters

Hindi UIT Mappings

Consonant mappings

Sr.	Hindi	UIT	Sr.	Hindi	UIT	Sr.	Hindi	UIT	Sr.	Hindi	UIT
1	क [k]	k	12	ठ [ʈʰ]	t`_h	23	ब [b]	b	34	क़ [q]	q
2	ख [kʰ]	k_h	13	ड [ɖ]	d`	24	भ [bʰ]	b_h	35	ख़ [x]	x
3	ग [g]	g	14	ढ [ɖʰ]	d`_h	25	म [m]	m	36	ग़ [ɣ]	X
4	घ [gʰ]	g_h	15	ण [ɳ]	n`	26	य [j]	j	37	ज़ [z]	z
5	ङ [ŋ]	N1	16	त [t̪]	t_d	27	र [r]	r	38	ड़ [ɽ]	r`
6	च [ʧ]	t_S	17	थ [t̪ʰ]	t_d_h	28	ल [l]	l	39	ढ़ [ɽʰ]	r`_h

7	छ [ʧʰ]	t_S_h	18	द [d̪]	d_d	29	व [v]	v	40	फ़ [f]	F
8	ज [ʤ]	d_Z	19	ध [d̪ʰ]	d_d_h	30	श [ʃ]	S	41	़	.
9	झ [ʤʰ]	d_Z_h	20	न [n]	n	31	ष [ʃ]	S1			
10	ञ [ŋʤ]	N2	21	प [p]	p	32	स [s]	s			
11	ट [ʈ]	t`	22	फ [pʰ]	p_h	33	ह [h]	h			

Table 80: UIT mappings of Hindi/Devanagari consonants

Vowel mappings

Sr.	Hindi	UIT	Sr.	Hindi	UIT	Sr.	Hindi	UIT	Sr.	Hindi	UIT
1	अ [ə]	@	7	ऋ [r̥]	r1	13	आँ [ɑ̃]	A~	19	ऍ [æ̃]	{~
2	आ [ɑ]	A	8	ए [e]	e	14	इँ [ĩ]	I~	20	ओं [õ]	o~
3	इ [ɪ]	I	9	ऐ [æ]	{	15	□ [ĩ]	i~	21	औं [ɔ̃]	O~
4	ई [i]	i	10	ओ [o]	o	16	उँ [ʊ̃]	U~			
5	उ [ʊ]	U	11	औ [ɔ]	O	17	ऊँ [ũ]	u~			
6	ऊ [u]	u	12	अँ [ə̃]	@~	18	एँ [ẽ]	e~			

Table 81: UIT mappings of Hindi vowel characters

Sr.	Hindi	UIT	Sr.	Hindi	UIT	Sr.	Hindi	UIT	Sr.	Hindi	UIT
1	ा [ɑ]	A1	6	ृ [r̥]	r2	11	ॅ [ə̃]	@~	16	ूॅ [ũ]	u~
2	ि [ɪ]	I	7	े [e]	e	12	ाॅ [ɑ̃]	A1~	17	ेॅ [ẽ]	e~
3	ी [i]	i	8	ै [æ]	{	13	िं [ĩ]	I~	18	ैं [æ̃]	{~
4	ु [ʊ]	U	9	ो [o]	o	14	ीं [ĩ]	i~	19	ों [õ]	o~
5	ू [u]	u	10	ौ [ɔ]	O	15	ुॅ [ʊ̃]	U~	20	ौं [ɔ̃]	O~

Table 82: UIT mappings of Hindi vowel signs

Annex 6 Finite-state Transducers

Hindi to UIT Finite-state Transducer

```
clear stack
set char-encoding UTF-8
!***************************************************
! START
!***************************************************
! ******************
! Definition of vairables
! 40 consonants कखगघङचछजझञटठडढणतथदधनपफबभमयरलवशषसहक़ख़ग़ज़ड़ढ़फ़
define                                                    CONSONANTS
[क|ख|ग|घ|ङ|च|छ|ज|झ|ञ|ट|ठ|ड|ढ|ण|त|थ|द|ध|न|प|फ|ब|भ|म|य|र|ल|व|श|ष|
स|ह|क़|ख़|ग़|ज़|ड़|ढ़|फ़];
define HINUMBER [०|१|२|३|४|५|६|७|८|९];
define ENGNUMBER [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9];
define HINUMBEREXPRESSION [ [HINUMBER [HINUMBER]*] ];
define ENGNUMBEREXPRESSION [ [ENGNUMBER [ENGNUMBER]*] ];
!******************
! Rules for Hindi to UIT Conversion
!******************
! ******************
! Other Symbols
read regex [◌़ -> ".", ◌ः -> H];
!******************
! Rules for Consonants
!******************
! Simple consonants
read regex [क -> k, ख -> [k "_" h], ग -> g, घ -> [g "_" h], ङ -> [N 1], च
-> [t "_" S], छ -> [t "_" S "_" h], ज -> [d "_" Z], झ -> [d "_" Z "_"
h], ञ -> [N 2], ट -> [t "`"], ठ -> [t "`" "_" h], ड -> [d "`"], ढ -> [d
"`" "_" h], ण -> [n "`"], त -> [t "_" d], थ -> [t "_" d "_" h], द -> [d
"_" d], ध -> [d "_" d "_" h], न -> n, प -> p, फ -> [p "_" h], ब -> b, भ ->
[b "_" h], म -> m, य -> j, र -> r, ल -> l, व -> v, श -> S, ष -> [S 1], स ->
s, ह -> h, क़ -> q, ख़ -> x, ग़ -> G, ज़ -> z, ड़ -> [r "`"], ढ़ -> [r "`" "_"
h], फ़ -> f];
! Other Aspirated Consonants
read regex [[◌् ह] -> ["_" h] || [र | ल | म | न] _ ];
! Special case for YA
read regex [य -> [j h] || _ [.#. | " "]];
!******************
! Rules for Nasalization
!******************
read regex [◌ँ -> "~", ◌ं -> "~"];
```

```
!*******************
! Rules for Vowels
!*******************
read regex [ऍ -> [I 2 e 1], ऐ -> [I 2 e 1], ऑ -> [I 2 o 1], ओ -> [I 2 o
1]];
! Dependent vowel sings that are very rare in thier use
read regex [ॅ -> [e 1], ॆ -> [e 1], ॉ -> [o 1], ॊ -> [o 1]];
! Dependent vowel signs that are very common in their use
read regex [ा -> [A 1], ि -> [I 1], ी -> [i 1], ु -> [U 1], ू -> [u 1], ृ ->
[r 1 1], े -> [e 1], ै -> ["{" 1], ो -> [o 1], ौ -> [O 1]];
! [ि उ] -> [I 1 j o 1]
read regex [अ -> "?", आ -> ["?" A 1], इ -> [I 2], उ -> [I 2 U 1], ई -> [I
2 i 1], [ऊ] -> [I 2 u 1], ए -> [I 2 e 1], [ऐ] -> [I 2 "{" 1], ओ -> [I 2 o
1], [औ] -> [I 2 O 1], ऋ -> [r 1]];
! when vowel sign U comes after a consonant and before the independent
vowel signs at the end of a word
read regex [[ु] -> [u 1] || CONSONANTS _ [[आ | ई | ए | ॅ | ं] [.#. | "
"]]];
! when vowels A, I & u come in the middle or at the end of a word af-
ter a vowel or nasalization
read regex [आ -> [A 1], ओ -> [o 1] || [इ | ई | ओ | ा | ि | ी | ु | ू | ृ | े
| ै | ो | ौ | ॅ | ं | ॉ | ॊ | ॅ | े] _ ];
! special string at the start of a word n equivalent in Urdu is BEH +
ALEF + LAM
! read regex [[ब ि अ ल] -> [b I 1 A 1 l] || [.#. | " "] _ ];
! vowels at the start of a word
read regex [अ -> "@", आ -> A, इ -> I, ई -> i, उ -> U, ऊ -> u, ऋ -> [r 1],
ए -> e, ऐ -> "{", ओ -> o, औ -> O, ऍ -> e, ऐ -> e, ऑ-> o, ओ -> o || [.#. |
" "| ् | "-"] _ ];
read regex [[ु] -> [u 1], [ि] -> [i 1] || CONSONANTS _ [.#. | " "] ];
read regex [ए -> e || "-" _ "-" ];
!*******************
! Rules for specail strings
!*******************
read regex [[प ु र] -> [p u 1 r], [ा ब ा द] -> [A b A 1 d "_" d], [अ ल ् ल
ा ह] -> ["@" l "." l A 1 h] || [? - [" " | .#.]] _ [.#. | " "]];
! *******************
! special words
read regex [[अ ल ् ल ा ह] -> ["@" l "." l A 1 h], ॐ -> [O m], [न] -> [n A
1], [य े] -> [j h], [व ो] -> [v h], [व े] -> [v h] || [" " | .#.] _ [.#. |
" "]];
!*******************
! Rules for Numbers
!*******************
read regex [० -> "+HIZERO", १ -> "+HIONE", २ -> "+HITWO", ३ -> "+HITHREE",
४ -> "+HIFOUR", ५ -> "+HIFIVE", ६ -> "+HISIX", ७ -> "+HISEVEN", ८ ->
```

```
"+HIEIGHT", ९ -> "+HININE", । -> "+SSTBD", ॥ -> "+DSTBD", "॰" ->
"+HIABRIVATION"];
! Decimal separator
read regex ["+DOT" -> "+HIDECIMAL", "+COMMA" -> "+HITHSEP" || HINUMBE-
REXPRESSION _ HINUMBEREXPRESSION];
read regex ["0" -> "+ZERO", 1 -> "+ONE", 2 -> "+TWO", 3 -> "+THREE", 4
-> "+FOUR", 5 -> "+FIVE", 6 -> "+SIX", 7 -> "+SEVEN", 8 -> "+EIGHT", 9
-> "+NINE"];
read regex ["+DOT" -> ["+DECIMAL"], "+COMMA" -> "+THSEP"  || ENGNUMBE-
REXPRESSION _ ENGNUMBEREXPRESSION];
! ******************
! Marks
read regex ["." -> "+DOT", "?" -> "+QMARK", "," -> "+COMMA", ";" ->
"+SEMICOLON", ":" -> "+COLON", "%" -> "+PERCENT", "+NL" -> "+NL"];
! ******************
! Hindi text normalization rules
! Removing all dot below from the text as they have been incorrectly
or mistakenly put in the text.
read regex [़ -> 0];
! normalizing consonants with a dot below
read regex [[क ़] -> क़, [ख ़] -> ख़,  [ग ़] -> ग़, [ज ़] -> ज़, [ड ़] -> ड़,
[ढ ़] -> ढ़, [फ ़] -> फ़];
!*************************************************
! END
!*************************************************
compose net
```

UIT to Hindi Finite-state Transducer

```
clear stack
set char-encoding UTF-8
!******************
!Definition of Varibales
!******************
define CONSONANTS [b | [b "_" h] | p | [p "_" h] | [t "_" d] | [t "_"
d "_" h] | [t "`"] | [t "`" "_" h] | [s 1] | [d "_" Z] | [d "_" Z "_"
h] | [t "_" S] | [t "_" S "_" h] | [h 1] | x | [d "_" d] | [d "_" d
"_" h] | [d "`"] | [d "`" "_" h] | [z 1] | r | [r "_" h] | [r "`"] |
[r "`" "_" h] | z | Z | s | S | [s 2] | [z 2] | [t "_" d 1] | [z 3] |
G | f | q | k | [k "_" h] | g | [g "_" h] | l | [l "_" h] | m | [m "_"
h] | n | [n "_" h] | v | [v "_" h] | h | j | [t "_" d 2] | H | [N 1] |
[N 2] | [n "`"] | [l "`"] | [b 1] | [d "_" Z 1] | [d "`" 1] | [g 1]];
define VOWELS ["@" | A | I | U | e | "{" | i | u | O | o | [A 1] | [A
2] | [A 3] | [I 1] | [I 2] | [U 1] | [e 1] | ["{" 1] | [i 1] | [u 1] |
[O 1] | [o 1] | [e 3] | [e 4] | ["{" 3] | ["{" 4] | [i 2] | [u 2]];
!******************
! Multi-character symbols
!******************
read regex ["+DOT" -> ".", "+QMARK" -> "?", "+DECIMAL" -> ".",
"+COMMA" -> ",", "+THSEP" -> ",", "+SEMICOLON" -> ";", "+COLON" ->
":", "+PERCENT" -> "%", "+ZERO" -> "0", "+ONE" -> 1, "+TWO" -> 2,
"+THREE" -> 3, "+FOUR" -> 4, "+FIVE" -> 5, "+SIX" -> 6, "+SEVEN" -> 7,
"+EIGHT" -> 8, "+NINE" -> 9, "+DSTBD" -> ॥, "+SSTBD" -> ।,
"+HIDECIMAL" -> ".", "+HITHSEP" -> ",", "+HIZERO" -> ०, "+HIONE" -> १,
"+HITWO" -> २, "+HITHREE" -> ३, "+HIFOUR" -> ४, "+HIFIVE" -> ५,
```

```
"+HISIX" -> ६, "+HISEVEN" -> ७, "+HIEIGHT" -> ८, "+HININE" -> ९,
"+HIABRIVATION" -> ॰];
!*******************
! Diacritics
!*******************
read regex ["." -> ◌्];
!*******************
! Consonants
!*******************
! Simple Consonants
read regex [b -> ब, p -> प, [t "_" d] -> त, [t "`"] -> ट, x -> ख़, [d
"_" d] -> द, [d "`"] -> ड, r -> र, z -> ज़, Z -> ज़, s -> स, S -> श, G
-> ग़, f -> फ़, q -> क़, k -> क, g -> ग, l -> ल, m -> म, n -> न, v ->
व, h -> ह, j -> य, H -> ◌ः, [N 1] -> ङ, [N 2] -> ञ];
! Consonants Set 2
read regex [[s 1] -> स, [d "_" Z] -> ज, [t "_" S] -> च, [h 1] -> ह,
[z 1] -> ज़, [r "`"] -> ड़, [S 1] -> ष, [s 2] -> स, [z 2] -> ज़, [t "_"
d 1] -> त, [z 3] -> ज़, [t "_" d 2] -> त, [n "`"] -> ण, [l "`"] -> ल,
[b 1] -> ब, [d "`" 1] -> ड, [g 1] -> ग];
! Consonants Set 3
read regex [[d "_" Z 1] -> ज];
! Aspirated Consonants
read regex [ ["_" h] -> [◌्ह]];
read regex [[b "_" h] -> भ, [p "_" h] -> फ, [t "_" d "_" h] -> थ, [t
"`" "_" h] -> ठ, [d "_" Z "_" h] -> झ, [t "_" S "_" h] -> छ, [d "_" d
"_" h] -> ध, [d "`" "_" h] -> ढ, [r "`" "_" h] -> ढ़, [k "_" h] -> ख,
[g "_" h] -> घ];

! HEH at the end of a word as vowel A
read regex [h -> ◌ा || [CONSONANTS] _ [.#. | " "]];
! *******************
! Nazalization
read regex ["~" -> ◌ं];
read regex ["~" -> ◌ं || [CONSONANTS] _ ];
read regex [ ["~" 1] -> ◌ं];
!*******************
! Vowels
!*******************
! Vowel characters
read regex [A -> आ, "@" -> अ, I -> इ, U -> उ, i -> ई, o -> ओ, O ->
औ, u -> ऊ, e -> ए, "{" -> ऐ, [r 1] -> ऋ];
read regex [[e 3] -> ए, ["{" 3] -> ऐ];
! Vowel signs
```

```
read regex [[A 1] -> ◌ा, [I 1] -> ◌ि, [U 1] -> ◌ु, [i 1] -> ◌ी, [o 1] -> ◌ो, [O 1] -> ◌ौ, [u 1] -> ◌ू, [e 1] -> ◌े, ["{" 1] -> ◌ै, [e 4] -> ◌े, ["{" 4] -> ◌ै, [r 1 1] -> ◌ृ];
! ******************
! Special cases
! ******************
! Special cases of vowels
read regex [["@" n] -> न || [? - .#.] _ ];
read regex [[A 2] -> ◌ा, [A 3] -> ◌ा, [I 2] -> इ, [i 2] -> ◌ी, [u 2] -> ◌ू, ["@" n 1] -> न, [I 1 n 1] -> [◌ि न], [U 1 n 1] -> [◌ु न]];
read regex [[u 1] -> ◌ु || CONSONANTS _ [[A | i | e | "~"] .#.]];
! Vowel with Ain
read regex ["?" -> अ];
read regex ["?" -> ◌ा || CONSONANTS _ ];
read regex [["?" A 1] -> आ, [A 1 "?"] -> [◌ा इ]];
read regex [["@" "?"] -> ए, ["?" I 1] -> इ, ["?" U 1] -> उ, ["?" e 1] -> ए, ["?" i 1] -> ई, ["?" "{" 1] -> ऐ, ["?" o 1] -> ओ, ["?" O 1] -> औ, ["?" u 1] -> ऊ, ["?" e 4] -> ए, ["?" "{" 4] -> ऐ];
!read regex [["@" "?"] -> ए, ["?" I 1] -> इ, ["?" U 1] -> उ, ["?" e 1] -> ए, ["?" i 1] -> ई, ["?" "{" 1] -> ऐ, ["?" o 1] -> ओ, ["?" O 1] -> औ, ["?" u 1] -> ऊ, ["?" e 4] -> ए, ["?" "{" 4] -> ऐ || [.#. | " " | "."] _ ];
! Ain after the final i sound in the word
read regex [["?" i 1] -> [◌ा ई] || _ [.#. | " "]];
read regex [[A 1] -> आ, [I 1] -> इ, [U 1] -> उ, [i 1] -> ई, [o 1] -> ओ, [O 1] -> औ, [u 1] -> ऊ, [e 1] -> ए, ["{" 1] -> ऐ, [e 4] -> ए, ["{" 4] -> ऐ, [r 1 1] -> ऋ || VOWELS _ ];
read regex [[I 2 i 1] -> ई, [I 2 e 1] -> ए, [I 2 "{" 1] -> ऐ, [I 2 o 1] -> ओ, [I 2 u 1] -> ऊ, [I 2 U 1] -> उ, [I 2 O 1] -> औ, [I 2 e 4] -> ए];
! ******************
! Compound Words
read regex [["-" e "-"] -> ["-" ए "-"], ["-" e 1 "-"] -> ["-" ए "-"]];
!******************
! Special words
read regex [ ["@" l "." l A 1 h] -> [अ ल ् ल ◌ा ह], [p u 1 r] -> [प ◌ु र] || _ [.#. | " "]];
read regex [["@" l "." l A 1 h] -> [अ ल ् ल ◌ा ह], [v h] -> [व ◌ो], [j h] -> [य ◌े], [n A 1] -> न, [O m] -> ॐ || [.#. | " " ] _ [ .#. | " " ]];
! ******************
! Multi-character symbols
read regex ["+DOT" -> "+DOT", "+QMARK" -> "+QMARK", "+DECIMAL" -> "+DECIMAL", "+COMMA" -> "+COMMA", "+NL" -> "+NL", "+THSEP" -> "+THSEP", "+SEMICOLON" -> "+SEMICOLON", "+COLON" -> "+COLON",
```

```
"+PERCENT" -> "+PERCENT", "+ZERO" -> "+ZERO", "+ONE" -> "+ONE", "+TWO"
-> "+TWO", "+THREE" -> "+THREE", "+FOUR" -> "+FOUR", "+FIVE" ->
"+FIVE", "+SIX" -> "+SIX", "+SEVEN" -> "+SEVEN", "+EIGHT" -> "+EIGHT",
"+NINE" -> "+NINE", "+DSTBD" -> "+DSTBD", "+SSTBD" -> "+SSTBD",
"+GURDECIMAL" -> "+HIDECIMAL", "+GURTHSEP" -> "+HITHSEP", "+GURZERO" -
> "+HIZERO", "+GURONE" -> "+HIONE", "+GURTWO" -> "+HITWO", "+GURTHREE"
-> "+HITHREE", "+GURFOUR" -> "+HIFOUR", "+GURFIVE" -> "+HIFIVE",
"+GURSIX" -> "+HISIX", "+GURSEVEN" -> "+HISEVEN", "+GUREIGHT" ->
"+HIEIGHT", "+GURNINE" -> "+HININE", "+URQMARK" -> "+QMARK", "+URSTBD"
-> "+SSTBD", "+URCOMMA" -> "+COMMA", "+URPERCENT" -> "+PERCENT",
"+URSEMICOLON" -> "+SEMICOLON", "+URSTAR" -> "*", "+URDECIMAL" ->
"+HIDECIMAL", "+URTHSEP" -> "+HITHSEP", "+URZERO" -> "+HIZERO",
"+URONE" -> "+HIONE", "+URTWO" -> "+HITWO", "+URTHREE" -> "+HITHREE",
"+URFOUR" -> "+HIFOUR", "+URFIVE" -> "+HIFIVE", "+URSIX" -> "+HISIX",
"+URSEVEN" -> "+HISEVEN", "+UREIGHT" -> "+HIEIGHT", "+URNINE" ->
"+HININE", "+HIABRIVATION" -> "+HIABRIVATION", "+HIDECIMAL" ->
"+HIDECIMAL", "+HITHSEP" -> "+HITHSEP", "+HIZERO" -> "+HIZERO",
"+HIONE" -> "+HIONE", "+HITWO" -> "+HITWO", "+HITHREE" -> "+HITHREE",
"+HIFOUR" -> "+HIFOUR", "+HIFIVE" -> "+HIFIVE", "+HISIX" -> "+HISIX",
"+HISEVEN" -> "+HISEVEN", "+HIEIGHT" -> "+HIEIGHT", "+HININE" ->
"+HININE", "" -> 0];
compose net
```

Urdu to UIT Finite-state Transducer

```
clear stack
set char-encoding UTF-8
!*******************
!Definition of Varibales
! Simple Consonants
define CONSONANTS [ب|پ|ت|ٹ|ث|ج|چ|ح|خ|د|ڈ|ذ|ر|ڑ|ز|ژ|س|ش|ص|ض|ط|ظ|ع|غ|ف|ق|ک|گ|ل|م|ن|ں|ہ|ۂ|ۃ|و|ی];
! Aspirated Consonants
define ASPICONSONANTS [[ھ [ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن]];
! Aspirated Consonants with ZER
define ASPIZERCONSONANTS [[ھ ِ [ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن]];
! Aspirated Consonants with ZABAR
define ASPIZABARCONSONANTS [[ھ َ [ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن]];
! Aspiracted Consonants with PESH
define ASPIZERCONSONANTS [[ھ ُ [ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن]];
! Aspirated Consonants with SHAD + ZER
define ASPISHADZERCONSONANTS [[ھ ِ ّ [ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن]];
! Aspirated Consonants with SHAD + ZABAR
define ASPISHADZABARCONSONANTS [[ھ َ ّ [ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن]];
! Aspiracted Consonants with SHAD + PESH
define ASPISHADZERCONSONANTS [[ھ ُ ّ [ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن]];
define ENGNUMBER [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9];
define URDNUMBER [۰ | ۱ | ۲ | ۳ | ۴ | ۵ | ۶ | ۷ | ۸ | ۹];
!define ENGNUMEXP [[([ "+" | "-" ])] [ [ENGNUMBER [ENGNUMBER | ","]*]
["." ENGNUMBER*]^<2 ]];
!define URDNUMEXP [[([ "+" | "-" ])] [ [URDNUMBER [URDNUMBER | ،]*] [،
URDNUMBER*]^<2 ]];
define ENGNUMBEREXPRESSION [ [ENGNUMBER [ENGNUMBER]*] ];
define URNUMBEREXPRESSION [ [URDNUMBER [URDNUMBER]*] ];
!*******************
! Diacritics
!*******************
! here we will ignore Mada Sign, Hamza Above, Hamza Below, and Noong-
huna Mark diacritics
```

```
read regex [ً -> 0, ٖ -> [I 1], ُ -> [U 1], ٰ -> [A 3], ِ -> [i 2], ٗ ->
[u 2], ٍ -> ["@" n 1], ٖ -> [I 1 n 1], ٌ","."<-ْ -> [U 1 n 1],  ,0 <-ٔ,0 <-ٕ
0 <-ٓ,0 <-ٖ];
! ********************
! Consonants
! Simple Consonants
read regex [ ب -> b, پ -> p, ت -> [t "_" d], ٹ -> [t "`"], ث -> [s
1], ج -> [d "_" Z], چ -> [t "_" S], ح -> [h 1], خ -> [x], د -> [d "_"
d], ڈ -> [d "`"], ذ -> [z 1], ر -> r, ڑ -> [r "`"], ز -> z, ژ -> Z, س
-> s, ش -> S, ص -> [s 2], ض -> [z 2], ط -> [t "_" d 1], ظ -> [z 3],
ف -> f, ق -> q, ک -> k, گ -> g, ل -> l, م -> m, و -> v, ہ -> h, ء ->
[I 2], ئ -> [I 2], ی -> j, غ -> G, ۃ -> [t "_" d 2], ھ -> h];
! Simple Consonants + SHAD
read regex [ [ ّ ب] -> [b "." b], [ ّ پ] -> [p "." p], [ ّ ت] -> [t "_" d
"." t "_" d], [ ّ ٹ] -> [t "`" "." t "`"], [ ّ ث] -> [s 1 "." s 1], [ ّ ج] -
> [d "_" Z "." d "_" Z], [ ّ چ] -> [t "_" S "." t "_" S], [ ّ ح] -> [h 1
"." h 1], [ ّ خ] -> [x "." x], [ ّ د] -> [d "_" d "." d "_" d], [ ّ ڈ] -> [d
"`" "." d "`"], [ ّ ذ] -> [z 1 "." z 1], [ ّ ر] -> [r "." r], [ ّ ڑ] -> [r
"`" "." r "`"], [ ّ ز] -> [z "." z], [ ّ ژ] -> [Z "." Z], [ ّ س] -> [s "."
s], [ ّ ش] -> [S "." S], [ ّ ص] -> [s 2 "." s 2], [ ّ ض] -> [z 2 "." z 2],
[ ّ ط] -> [t "_" d 1 "." t "_" d 1], [ ّ ظ] -> [z 3 "." z 3], [ ّ ف] -> [f
"." f], [ ّ ق] -> [q "." q], [ ّ ک] -> [k "." k], [ ّ گ] -> [g "." g], [ ّ ل]
-> [l "." l], [ ّ م] -> [m "." m], [ ّ ہ] -> [h "." h], [ ّ غ] -> [G "." G],
[ ّ ھ] -> [h "." h], [ ّ ۃ] -> [t "_" d 2 "." t "_" d 2]];
! Aspirated Consonants
read regex [ ھ -> ["_" h] || [ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن] _ ];
! Aspirated Consonants + ZABAR
read regex [ [ب َ ھ] -> [b "_" h], [پ َ ھ] -> [p "_" h], [ت َ ھ] -> [t "_"
d "_" h], [ٹ َ ھ] -> [t "`" "_" h], [ج َ ھ] -> [d "_" Z "_" h], [چ َ ھ] ->
[t "_" S "_" h], [د َ ھ] -> [d "_" d "_" h], [ڈ َ ھ] -> [d "`" "_" h], [ر َ
ھ] -> [r "_" h], [ڑ َ ھ] -> [r "`" "_" h], [ک َ ھ] -> [k "_" h], [گ َ ھ] ->
[g "_" h], [ل َ ھ] -> [l "_" h], [م َ ھ] -> [m "_" h], [ن َ ھ] -> [n "_"
h]];
! Aspirated Consonants + ZER
read regex [ [ب ِ ھ] -> [b "_" h I 1], [پ ِ ھ] -> [p "_" h I 1], [ت ِ ھ] ->
[t "_" d "_" h I 1], [ٹ ِ ھ] -> [t "`" "_" h I 1], [ج ِ ھ] -> [d "_" Z "_"
h I 1],  [چ ِ ھ] -> [t "_" S "_" h I 1], [د ِ ھ] -> [d "_" d "_" h I 1], [ڈ
ِ ھ] -> [d "`" "_" h I 1], [ر ِ ھ] -> [r "_" h I 1], [ڑ ِ ھ] -> [r "`" "_"
h I 1], [ک ِ ھ] -> [k "_" h I 1], [گ ِ ھ] -> [g "_" h I 1], [ل ِ ھ] -> [l
"_" h I 1], [م ِ ھ] -> [m "_" h I 1], [ن ِ ھ] -> [n "_" h I 1]];
! Aspirated Consonants + ZER to ignore ZER while transliteration when
followed by CHOTI_YEH
read regex [ [ب ِ ھ] -> [b "_" h], [پ ِ ھ] -> [p "_" h], [ت ِ ھ] -> [t "_"
d "_" h], [ٹ ِ ھ] -> [t "`" "_" h], [ج ِ ھ] -> [d "_" Z "_" h], [چ ِ ھ] ->
[t "_" S "_" h], [د ِ ھ] -> [d "_" d "_" h], [ڈ ِ ھ] -> [d "`" "_" h], [ر ِ
ھ] -> [r "_" h], [ڑ ِ ھ] -> [r "`" "_" h], [ک ِ ھ] -> [k "_" h], [گ ِ ھ] -
> [g "_" h], [ل ِ ھ] -> [l "_" h], [م ِ ھ] -> [m "_" h], [ن ِ ھ] -> [n "_"
h] || _ i];
! Aspirated Consonants + PESH
read regex [ [ب ُ ھ] -> [b "_" h U 1], [پ ُ ھ] -> [p "_" h U 1], [ت ُ ھ] ->
[t "_" d "_" h U 1], [ٹ ُ ھ] -> [t "`" "_" h U 1], [ج ُ ھ] -> [d "_" Z "_"
h U 1],  [چ ُ ھ] -> [t "_" S "_" h U 1], [د ُ ھ] -> [d "_" d "_" h U 1], [ڈ
ُ ھ] -> [d "`" "_" h U 1], [ر ُ ھ] -> [r "_" h U 1], [ڑ ُ ھ] -> [r "`" "_"
```

```
h U 1], [ھ ُ ک] -> [k "_" h U 1], [ھ ُ گ] -> [g "_" h U 1], [ھ ُ ل] -> [l
"_" h U 1], [ھ ُ م] -> [m "_" h U 1], [ھ ُ ن] -> [n "_" h U 1]];
! Aspirated Consonants + PESH to ignore PESH while transliteration
when followed by WAW
read regex [ [ھ ُ ب] -> [b "_" h], [ھ ُ پ] -> [p "_" h], [ھ ُ ت] -> [t "_"
d "_" h], [ھ ُ ٹ] -> [t "`" "_" h], [ھ ُ ج] -> [d "_" Z "_" h], [ھ ُ چ] ->
[t "_" S "_" h], [ھ ُ د] -> [d "_" d "_" h], [ھ ُ ڈ] -> [d "`" "_" h], [ ُ ر
ھ] -> [r "_" h], [ھ ُ ڑ] -> [r "`" "_" h], [ھ ُ ک] -> [k "_" h], [ھ ُ گ] -
> [g "_" h], [ھ ُ ل] -> [l "_" h], [ھ ُ م] -> [m "_" h], [ھ ُ ن] -> [n "_"
h] || _ u];
! Aspirated Consonants + SHAD
read regex [ [ھ ّ ب] -> [b "." b "_" h], [ھ ّ پ] -> [p "." p "_" h], [ ّ ت
ھ] -> [t "_" d "." t "_" d "_" h], [ھ ّ ٹ] -> [t "`" "." t "`" "_" h],
[ھ ّ ج] -> [d "_" Z "." d "_" Z "_" h], [ھ ّ چ] -> [t "_" S "." t "_" S
"_" h], [ھ ّ د] -> [d "_" d "." d "_" d "_" h], [ھ ّ ڈ] -> [d "`" "." d "`"
"_" h], [ھ ّ ر] -> [r "." r "_" h], [ھ ّ ڑ] -> [r "`" "." r "`" "_" h], [ک
ھ ّ] -> [k "." k "_" h], [ھ ّ گ] -> [g "." g "_" h], [ھ ّ ل] -> [l "." l
"_" h], [ھ ّ م] -> [m "." m "_" h], [ھ ّ ن] -> [n "." n "_" h],[ب ھ ّ] -> [b
"." b "_" h], [پ ھ ّ] -> [p "." p "_" h], [ت ھ ّ] -> [t "_" d "." t "_" d
"_" h], [ٹ ھ ّ] -> [t "`" "." t "`" "_" h], [ج ھ ّ] -> [d "_" Z "." d "_"
Z "_" h], [چ ھ ّ] -> [t "_" S "." t "_" S "_" h], [د ھ ّ] -> [d "_" d "."
d "_" d "_" h], [ڈ ھ ّ] -> [d "`" "." d "`" "_" h], [ر ھ ّ] -> [r "." r
"_" h], [ڑ ھ ّ] -> [r "`" "." r "`" "_" h], [ک ھ ّ] -> [k "." k "_" h],
[گ ھ ّ] -> [g "." g "_" h], [ل ھ ّ] -> [l "." l "_" h], [م ھ ّ] -> [m "." m
"_" h], [ن ھ ّ] -> [n "." n "_" h]];
! Aspirated Consonants + SHAD + ZABAR
read regex [ [ھ ّ َ ب] -> [b "." b "_" h], [ھ ّ َ پ] -> [p "." p "_" h], [ ت
ھ ّ َ] -> [t "_" d "." t "_" d "_" h], [ھ ّ َ ٹ] -> [t "`" "." t "`" "_"
h], [ھ ّ َ ج] -> [d "_" Z "." d "_" Z "_" h], [ھ ّ َ چ] -> [t "_" S "." t
"_" S "_" h], [ھ ّ َ د] -> [d "_" d "." d "_" d "_" h], [ھ ّ َ ڈ] -> [d "`"
"." d "`" "_" h], [ھ ّ َ ر] -> [r "." r "_" h], [ھ ّ َ ڑ] -> [r "`" "." r
"`" "_" h], [ھ ّ َ ک] -> [k "." k "_" h], [ھ ّ َ گ] -> [g "." g "_" h], [ ّ ل
ھ َ] -> [l "." l "_" h], [ھ ّ َ م] -> [m "." m "_" h], [ھ ّ َ ن] -> [n "." n
"_" h]];
! Aspirated Consonants + SHAD + ZER
read regex [ [ھ ّ ِ ب] -> [b "." b "_" h I 1], [ھ ّ ِ پ] -> [p "." p "_" h I
1], [ھ ّ ِ ت] -> [t "_" d "." t "_" d "_" h I 1], [ھ ّ ِ ٹ] -> [t "`" "." t
"`" "_" h I 1], [ھ ّ ِ ج] -> [d "_" Z "." d "_" Z "_" h I 1], [ھ ّ ِ چ] ->
[t "_" S "." t "_" S "_" h I 1], [ھ ّ ِ د] -> [d "_" d "." d "_" d "_" h I
1], [ھ ّ ِ ڈ] -> [d "`" "." d "`" "_" h I 1], [ھ ّ ِ ر] -> [r "." r "_" h I
1], [ھ ّ ِ ڑ] -> [r "`" "." r "`" "_" h I 1], [ھ ّ ِ ک] -> [k "." k "_" h I
1], [ھ ّ ِ گ] -> [g "." g "_" h I 1], [ھ ّ ِ ل] -> [l "." l "_" h I 1], [ ّ ِ م
ھ] -> [m "." m "_" h I 1], [ھ ّ ِ ن] -> [n "." n "_" h I 1]];
! Aspirated Consonants + SHAD + ZER while transliteration when fol-
lowed by CHOTI_YEH
read regex [ [ھ ّ ِ ب] -> [b "." b "_" h], [ھ ّ ِ پ] -> [p "." p "_" h], [ ت
ھ ّ ِ] -> [t "_" d "." t "_" d "_" h], [ھ ّ ِ ٹ] -> [t "`" "." t "`" "_"
h], [ھ ّ ِ ج] -> [d "_" Z "." d "_" Z "_" h], [ھ ّ ِ چ] -> [t "_" S "." t
"_" S "_" h], [ھ ّ ِ د] -> [d "_" d "." d "_" d "_" h], [ھ ّ ِ ڈ] -> [d "`"
"." d "`" "_" h], [ھ ّ ِ ر] -> [r "." r "_" h], [ھ ّ ِ ڑ] -> [r "`" "." r
"`" "_" h], [ھ ّ ِ ک] -> [k "." k "_" h], [ھ ّ ِ گ] -> [g "." g "_" h], [ ّ ل
ھ ِ] -> [l "." l "_" h], [ھ ّ ِ م] -> [m "." m "_" h], [ھ ّ ِ ن] -> [n "." n
"_" h] || _ i];
! Aspirated Consonants + SHAD + PESH
```

```
read regex [ [ب ّ ُ ھ] -> [b "." b "_" h U 1], [پ ّ ُ ھ] -> [p "." p "_" h U
1], [ت ّ ُ ھ] -> [t "_" d "." t "_" d "_" h U 1], [ٹ ّ ُ ھ] -> [t "`" "." t
"`" "_" h U 1], [ج ّ ُ ھ] -> [d "_" Z "." d "_" Z "_" h U 1], [چ ّ ُ ھ] ->
[t "_" S "." t "_" S "_" h U 1], [د ّ ُ ھ] -> [d "_" d "." d "_" d "_" h U
1], [ڈ ّ ُ ھ] -> [d "`" "." d "`" "_" h U 1], [ر ّ ُ ھ] -> [r "." r "_" h U
1], [ڑ ّ ُ ھ] -> [r "`" "." r "`" "_" h U 1], [ک ّ ُ ھ] -> [k "." k "_" h U
1], [گ ّ ُ ھ] -> [g "." g "_" h U 1], [ل ّ ُ ھ] -> [l "." l "_" h U 1], [م ّ ُ
ھ] -> [m "." m "_" h U 1], [ن ّ ُ ھ] -> [n "." n "_" h U 1]];
! Aspirated Consonants + SHAD + PESH while transliteration when fol-
lowed by WAW
read regex [ [ب ّ ُ ھ] -> [b "." b "_" h], [پ ّ ُ ھ] -> [p "." p "_" h], [ت
ّ ُ ھ] -> [t "_" d "." t "_" d "_" h], [ٹ ّ ُ ھ] -> [t "`" "." t "`" "_"
h], [ج ّ ُ ھ] -> [d "_" Z "." d "_" Z "_" h], [چ ّ ُ ھ] -> [t "_" S "." t
"_" S "_" h], [د ّ ُ ھ] -> [d "_" d "." d "_" d "_" h], [ڈ ّ ُ ھ] -> [d "`"
"." d "`" "_" h], [ر ّ ُ ھ] -> [r "." r "_" h], [ڑ ّ ُ ھ] -> [r "`" "." r
"`" "_" h], [ک ّ ُ ھ] -> [k "." k "_" h], [گ ّ ُ ھ] -> [g "." g "_" h], [ل ّ
ُ ھ] -> [l "." l "_" h], [م ّ ُ ھ] -> [m "." m "_" h], [ن ّ ُ ھ] -> [n "." n
"_" h] || _ u];
!*******************
! Gol Heh at the end of a word
! Gol Heh is considered as a vowel A1 when it is at the end of a word
and is preceeded by a consonant. we will not convert it into A1, as
when we need to convert the text back in Shahmukhi, it will produce
ambiguity. we will handle this issue when we will convert UIT into
Gurmukhi. Two Gol Hehs at the end of word are considered as consonant
'h'. Or a Gol heh is considered as consonant 'h' when it is at the end
of word and is preceeded by a vowel.
! *******************
! Short Vowels with consonants
! WAW as vowel o after a consonant
read regex [و -> [o 1] || [CONSONANTS | ASPICONSONANTS | ASPISHADCONSO-
NANTS | j | v | n] _ ];
! ZABAR + WAW as au after a consonant
read regex [[ َ و] -> [O 1] || [CONSONANTS | ASPICONSONANTS | ASPISHAD-
CONSONANTS | j | v | n] _ ];
! PESH + WAW as u after a consonant
read regex [[ ُ و] -> [u 1] || [CONSONANTS | ASPICONSONANTS | ASPISHAD-
CONSONANTS | j | v | n] _ ];
! WAW as au after a consonant
read regex [و -> [O 1] || [ASPIZABARCONSONANTS | ASPISHADZABARCONSO-
NANTS] _ ];
! WAW as u after a consonant
read regex [و -> [u 1] || [ASPIPESHCONSONANTS | ASPISHADPESHCONSONANTS]
_ ];
! YEH as vowel e after a consonant
read regex [ ی -> [e 1] || [CONSONANTS | ASPICONSONANTS | ASPISHADCON-
SONANTS | j | v | n] _ ];
! ZABAR + YEH as vowel ai after a consonant
read regex [[ َ 1 "}"] <- [ی] || [CONSONANTS | ASPICONSONANTS | ASPISHAD-
CONSONANTS | j | v | n] _ ];
! ZER + YEH as vowel i after a consonant
read regex [[ ِ ی] -> [i 1] || [CONSONANTS | ASPICONSONANTS | ASPISHAD-
CONSONANTS | j | v | n] _ ];
! YEH as i after a consonant
read regex [ی -> [i 1] || [ASPIZERCONSONANTS | ASPISHADZERCONSONANTS]
_ ];
! YEH as i after a consonant
```

```
read regex [1 "}"] <- ى] || [ASPIZABARCONSONANTS | ASPISHADZABARCONSO-
NANTS] _ ];
!******************
! Long Vowels not at the start of a word
! ******************
! Cases of Bari YEH
! Bari Yeh not at the end of a word
read regex [ے -> [e 4]];
! Bari YEH after ZABAR and not at the end of a word
read regex [ َ 4 "}"] <- ے]];
read regex [[ےء] -> [I 2 e 4], [ےئ] -> [I 2 e 4]];
read regex [4 "}"] <- ے] || [ASPIZABARCONSONANTS | ASPISHADZABARCONSO-
NANTS] _ ];
! Bari yeh will always form the vowel with a number 4 in the middle or
at the end of a word
! Vowels with AIN any where else
read regex [ ع -> ["?"] ];
read regex [ [َع] -> ["?"] ];
read regex [ [ِع] -> ["?" I 1], [ُع] -> ["?" U 1]];
read regex [ [ىع] -> ["?" e 1], [ىِع] -> ["?" i 1], [ ,[1 "}" "?"] <- [ىَع
وع]] -> ["?" o 1], [ وَع] -> ["?" O 1], [وُع] -> ["?" u 1], [ےع] ->
["?" e 4], [4 "}" "?"] <- [ےَع]];
! Vowels with ALEF any where else
read regex [ آ -> A];
read regex [ ا -> [A 1] ];
read regex [ [َا] -> [A 1] ];
read regex [ [ِا] -> I, [ُا] -> U];
read regex [ [ىا] -> e, [ىِا] -> i, [وا],"}" <- [ىَا] -> o, [ وَا] -> O, [ا
وُ] -> u, [ےا] -> [e 3], [3 "}"] <- [ےَا]];
! Vowels with AIN at the start of a word or a syllable
read regex [ ْ|.#.] || "?" <- ع | " " | "-"] _ ];
read regex [ [ْ|.#.] || "?" <- [َع | " " | "-"] _ ];
read regex [ [ِع] -> ["?" I 1], [ُع] -> ["?" U 1] || [.#. | ْ | " " |
"-"] _ ];
read regex [ [ىع] -> ["?" e 1], [ىِع] -> ["?" i 1], [ ,[1 "}" "?"] <- [ىَع
وع]] -> ["?" o 1], [ وَع] -> ["?" O 1], [وُع] -> ["?" u 1], [ےع] ->
["?" e 4], [ْ|.#.] || [4 "}" "?"] <- [ےَع | " " | "-"] _ ];
! Vowels with ALEF at the start of a word or a syllable
read regex [ آ -> A || [.#. | ْ | " " | "-"] _ ];
read regex [ ْ|.#.] || ["@"] <- ا | " " | "-"] _ ];
read regex [ [ْ|.#.] || ["@"] <- [َا | " " | "-"] _ ];
read regex [ [ِا] -> I, [ُا] -> U || [.#. | ْ | " " | "-"] _ ];
read regex [ [ىا] -> e, [ىِا] -> i, [وا],"}" <- [ىَا] -> o, [ وَا] -> O, [ا
وُ] -> u, [ےا] -> [e 3], [ْ|.#.] || [3 "}"] <- [ےَا | " " | "-"] _ ];
! ******************
! HAMZA after YEH as "@" vowel sound
read regex [[ىئ] -> [j I 2], [ىء] -> [j I 2] || [CONSONANTS | ASPI-
CONSONANTS | ASPISHADCONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSO-
NANTS | ASPISHADZABARCONSONANTS | ASPISHADPESHCONSONANTS | ASPIZERCON-
SONANTS | ASPISHADZERCONSONANTS | j | v | n] _ [CONSONANTS | ASPICON-
SONANTS | ASPISHADCONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSO-
NANTS | ASPISHADZABARCONSONANTS | ASPISHADPESHCONSONANTS | ASPIZERCON-
SONANTS | ASPISHADZERCONSONANTS | j | v | n]];
! ******************
! NOONGHUNA
read regex [ں -> "~"];
```

```
! NOON as a nasalization
read regex [ن -> "~"];
! Noon followed by Noonghuna mark
read regex [[ٞ ن] -> "~"];
! ********************
! NOON when it is not considered as NOONGHUNA
read regex [[ ّن] -> [n "." n]];
! NOON after ALEF, AIN, ALEF Mada, Bari YEH
!read regex [ن -> n || [ ع|ے|آ|ا ] _ ];
! NOON before ALEF, ALEF Mada, Bari YEH, and some diacritics
read regex [ن -> n || _ [ٰ|ٗ|ٔ|ٕ|ٖ|ٙ|ٚ|ً|ٌ|ٍ|َ|ُ|ِ|ّ|ْ|ۃ|ہ|ھ|ۂ|ۓ|ۓ|ں|و|ی|ن|ع|ے|آ|ا | [i 1] | ئ|ء | v | j | I | ["@" n] | A] ];
! NOON at the end of a word
read regex [ن -> n || _ [ہ [.#. | " "]] ];
read regex [ن -> n || _ [ .#. | ٔ | " " | "-"] ];
! NOON at the start of a word
read regex [ن -> n || [ .#. | ٔ | " " | "-"] _ ];
! ********************
read regex [و -> v, ی -> j || [ [? - [.#. | " "]] [آ|ا]] _ ];
read regex [و -> [o 1] || [[I 2 i 1] | [I 2 e 1] | [I] | [I 1]] _ ];
! Special words with YEH + WAW
read regex [و -> v || ی _ ];
! YEH preceeded by a consonant and followed by WAW فرکیوں، بیماریوں، خرابیوں، فرکیاں، بیماریاں، خرابیاں
read regex [ی -> [I 1 j] || [CONSONANTS | ASPICONSONANTS | ASPISHAD-CONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPISHADZABAR-CONSONANTS | ASPISHADPESHCONSONANTS | ASPIZERCONSONANTS | ASPISHADZER-CONSONANTS] _ [ [ں[و|ا [.#. | " "]]];
! Consonant + YEH + WAW + Consonant at the start of the word
read regex [[و ی] -> [e 1 v] || [[.#. |ٔ | " "] [CONSONANTS | ASPICON-SONANTS | ASPISHADCONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSO-NANTS | ASPISHADZABARCONSONANTS | ASPISHADPESHCONSONANTS | ASPIZERCON-SONANTS | ASPISHADZERCONSONANTS]] _ [CONSONANTS | ASPICONSONANTS | AS-PISHADCONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPI-SHADZABARCONSONANTS | ASPISHADPESHCONSONANTS | ASPIZERCONSONANTS | AS-PISHADZERCONSONANTS]];
! YEH + WAW at the end of a word
read regex [و -> [o] || [[CONSONANTS | ASPICONSONANTS]ٔ| .#.] ں] _ [ی ٖ | " "]]];
read regex [[وی] -> [j o 1] || _ [.#. |ٔ | " "]];
! ********************
! WAW as Consonants
! YEH and WAW before NOON at the end of a word
read regex [و -> v || _ [[ہ] [ .#. | " " ]]];
! WAW after ZER
read regex [و -> v || [[[CONSONANTS | ASPICONSONANTS] ٖ] | [[CONSONANTS | ASPICONSONANTS] ٖ ّ]] _ ];
read regex [[ ّو] -> [v "." v]];
! Waw is followed by a vowel
read regex [و -> v || _ [ٰ|ٗ|ٔ|ٕ|ٖ|ٙ|ٚ|ً|ٌ|ٍ|َ|ُ|ِ|ّ|ْ|ے|ی|آ|ا | [i 1]]];
read regex [و -> v || [و|ی|آ|ع|ا] _ [ع|آ|و|ی|ا]];
! Waw at the start of a word
read regex [و -> v || [.#. |ٔ | " " | "-"] _ ];
! ********************
! YEH and WAW at the end after ALEF, ALEF mada, HAMZA + YEH, HAMZA + Bari YEH at the end of a word
! will create problem with the word hvA1 (air)
```

```
read regex [و -> [U 1] || [آ|ا]] _ [ه | [I 2 i 1] | [I 2 e 1] | [I 2 i 1
ں] | [I 2 e 1 ْ|.#.] [[ں | " " | "-"]]];
!*******************
! When Yeh will be considered as a consonant
read regex [[ّی] -> [j "." j]];
! YEH after PESH
read regex [ی -> j || [[[CONSONANTS | ASPICONSONANTS | ASPISHADCONSO-
NANTS | ASPIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPISHADZABARCONSO-
NANTS | ASPISHADPESHCONSONANTS | ASPIZERCONSONANTS | ASPISHADZERCONSO-
NANTS | ُ [و] | [[CONSONANTS | ASPICONSONANTS | ASPISHADCONSONANTS | AS-
PIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPISHADZABARCONSONANTS | AS-
PISHADPESHCONSONANTS | ASPIZERCONSONANTS | ASPISHADZERCONSONANTS | ّ [و
ُ]] _ ];
read regex [[ّ ی] -> [i 1 j] || [CONSONANTS | ASPICONSONANTS | ASPI-
SHADCONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPISHAD-
ZABARCONSONANTS | ASPISHADPESHCONSONANTS] _ ];
read regex [[ّ ی] -> [i 1 j] || [ASPIZERCONSONANTS | ASPISHADZERCONSO-
NANTS] _ ];
read regex [[ِّ ی] -> [i 1 j] || [CONSONANTS | ASPICONSONANTS | ASPI-
SHADCONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPISHAD-
ZABARCONSONANTS | ASPISHADPESHCONSONANTS] _ ];
! Yeh is followed by a diacritic or a vowel characters
read regex [ی -> j || _ [ع|ے|آ|ا|ٰ|ۃ|ْ|ِ|ُ|َ|ٍ|ٌ|ً | [i 1]]];
! YEH and WAW before NOON at the end of a word
read regex [ی -> j || _ [[ ہ ] [ .#. | " " ]]];
! Yeh in between of two Alefs
! ایام، عیال، عیاں
read regex [ی -> j || [و|ی|آ|ع|ا] _ [ی|و|آ|ع|ا]];
! Yeh followed by Shad after Alef or Ain
read regex [[ّی] -> [j "." j] || [ع|ا] _ ];
! اعیان
read regex [ی -> j || [[.#. | ا] _ [[ع ا] ["-"|" "|ْ]];
! YEH at start of a word
read regex [ی -> j || [.#. | ْ | " " | "-"] _ ];
! YEH at the end of a word
read regex [ی -> [i 1] || _ [.#. | ْ | " " | "-"]];
! *******************
! Special words with HAMZA + WAW
! HAMZA + WAW (may have ambiguity)
read regex [ [ئو] -> [I 2 o 1], [ءو] -> [I 2 o 1], [ئ ُو] -> [I 2 u 1],
[ء ُو] -> [I 2 u 1]];
! *******************
! Special words with HAMZA + YEH
! HAMZA + YEH -> [I 2 e 1] after a vowel

read regex [[ئی] -> [I 2 e 1], [ءی] -> [I 2 e 1]];
! HAMZA + YEH -> [I 2 i 1] after a Consonant
read regex [[ئ ی] -> [I 2 i 1], [ء ی] -> [I 2 i 1] || [CONSONANTS | AS-
PICONSONANTS | ASPISHADCONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCON-
SONANTS | ASPISHADZABARCONSONANTS | ASPISHADPESHCONSONANTS | ASPIZER-
CONSONANTS | ASPISHADZERCONSONANTS | ی|و] _ ];

! HAMZA + YEH any where else
read regex [[ئ ِی] -> [I 2 i 1], [ء ِی] -> [I 2 i 1]];
! *******************
! Special strings at the end of a word
```

```
read regex [ [ٰ ی] -> A 2, [ٰ ی] -> A 2, [ً ا] -> ["@" n], [ی ء] -> [I 2 i
1], [ی ئ] -> [I 2 i 1], [و ء] -> [I 2 o 1], [و ئ] -> [I 2 o 1] || _ [.#.
| ٔ | " " | "-"] ];
! ******************
! KAF + YEH + WAW + NOON + KAF + HEHGOL, CHEH + WAW + NOON + KAF +
HEHGOL
! when kI1 comes at the end of some words
read regex [[ه ک] -> [k I 1] || [[.#. | _ [[[ل ب] |[ا ن] | [و ج] | [ب ج]] ["-" | " " | ٔ
ٔ|.#.] | " " | "-"]];
! YEH + WAW + NOONGHUNA and KAF + YEH + WAW + NOONGHUNA as words and
words ending in K I 1
read regex [[ں و ی] -> [j u 1 "~"], [ں و ی ک] -> [k j u 1 "~"], [ک ی و ن ک
ہ] -> [k j u 1 "~" k I 1], [ہ ک ن و چ] -> [t "_" S u 1 "~" k I 1], [ک ی ُو
ں] -> [k j u 1 "~"], [ہ ک ن و ُ ی ک] -> [k j u 1 "~" k I 1], [ہ ک ن و ُ چ] ->
[t "_" S u 1 "~" k I 1] || [ .#. | " " ] _ [ .#. | " " ]];
! ******************
! Compound Words
!read regex [ ء -> ["-" e "-"], ء -> ["-" e  1 "-"] || [CONSONANTS |
ASPICONSONANTS] _ [" "]];
read regex [ [ء " "] -> ["-" e "-"] || [CONSONANTS | ASPICONSONANTS] _
];
! ******************
! Vowels only
read regex [[ٔ|.#.] _ ["-"|""|ٔ|.#.] || "@" <- [ا | " " | "-"]];
read regex [[ٕ],"@" <- [ٖ ا] -> I, [ُ ا] -> U || [.#. | ٔ|.#.] _ ["-"|""|ٔ | " "
| "-"]];
read regex [[ا ی] -> i, [ا ٖ ی] -> i, [ا و],["}"] <- [ا ٖ ی] -> o, [ا ٖ و] -> O, [ا
ُ و] -> u, [ا ے] -> [e 3], [ٔ|.#.] _ ["-"|""|ٔ|.#.] || [3 "}"] <- [ا ٖ ے | " " | "-"]
];
! ******************
! Special Words
read regex [[ﷲ] -> ["@" l "." l A 1 h]];
read regex [[ا ﷲ] -> ["@" l "." l A 1 h], [ہ ل ل ا] -> ["@" l "." l A 1
h], [ہ ٰ ّ ل ل ا] -> ["@" l "." l A 1 h], [ہ ّ ل ل ا] -> ["@" l "." l A 1 h]];
read regex [[ہ ک] -> [k I 1], [ہ و] -> [v h], [ہ ی] -> [j h] || [ .#. | "
" ] _ [ .#. | " " ]];
!******************
! Number
!******************
read regex [۰ -> "+URZERO", ۱ -> "+URONE", ۲ -> "+URTWO", ۳ ->
"+URTHREE", ۴ -> "+URFOUR", ۵ -> "+URFIVE", ۶ -> "+URSIX", ۷ ->
"+URSEVEN", ۸ -> "+UREIGHT", ۹ -> "+URNINE"];
! Decimal and thousand separator
read regex [٫ -> ["+URDECIMAL"], ٬ -> "+URTHSEP"];
read regex ["+DOT" -> "+URDECIMAL", "+COMMA" -> "+URTHSEP" || URNUMBE-
REXPRESSION _ URNUMBEREXPRESSION];
read regex ["0" -> "+ZERO", 1 -> "+ONE", 2 -> "+TWO", 3 -> "+THREE", 4
-> "+FOUR", 5 -> "+FIVE", 6 -> "+SIX", 7 -> "+SEVEN", 8 -> "+EIGHT", 9
-> "+NINE", ";" -> "+SEMICOLON", ":" -> "+COLON", "%" -> "+PERCENT"];
read regex ["+DOT" -> "+DECIMAL", "+COMMA" -> "+THSEP" || ENGNUMBEREX-
PRESSION _ ENGNUMBEREXPRESSION];
read regex ["." -> "+DOT", ؟ -> "+URQMARK", ۔ -> "+URSTBD", ،-> 
"+URCOMMA", "," -> "+COMMA", "+NL" -> "+NL", ٪-> "+URPERCENT", ؛ ->
"+URSEMICOLON", * -> "+URSTAR", "" -> 0];
compose net
```

UIT to Urdu Finite-state Transducer

```
clear stack
set char-encoding UTF-8
! ******************
! Definitions of variables
define CONSONANTS [b | [b "_" h] | p | [p "_" h] | [t "_" d] | [t "_"
d "_" h] | [t "`"] | [t "`" "_" h] | [s 1] | [d "_" Z] | [d "_" Z "_"
h] | [t "_" S] | [t "_" S "_" h] | [h 1] | x | [d "_" d] | [d "_" d
"_" h] | [d "`"] | [d "`" "_" h] | [z 1] | r | [r "_" h] | [r "`"] |
[r "`" "_" h] | z | Z| s | S | [S 1] | [s 2] | [z 2] | [t "_" d 1] |
[z 3] | G | f | q | k | [k "_" h] | g | [g "_" h] | l | [l "_" h] | m
| [m "_" h] | n | [n "_" h] | v | [v "_" h] | h | j | [t "_" d 2] | H
| [N 1] | [N 2] | [n "`"] | [l "`"] | [b 1] | [d "_" Z 1] | [d "`" 1]
| [g 1]];
define VOWELS ["@" | A | I | U | e | "{" | i | u | O | o | [A 1] | [A
2] | [A 3] | [I 1] | [I 2] | [U 1] | [e 1] | ["{" 1] | [i 1] | [u 1] |
[O 1] | [o 1] | [e 3] | [e 4] | ["{" 3] | ["{" 4] | [i 2] | [u 2]];
!******************
! Multi-character symbols
!******************
read regex ["+DOT" -> ".", "+QMARK" -> "?", "+DECIMAL" -> ".",
"+COMMA" -> ",", "+THSEP" -> ",", "+SEMICOLON" -> ";", "+COLON" ->
":", "+PERCENT" -> ٪, "+ZERO" -> "0", "+ONE" -> 1, "+TWO" -> 2,
"+THREE" -> 3, "+FOUR" -> 4, "+FIVE" -> 5, "+SIX" -> 6, "+SEVEN" -> 7,
"+EIGHT" -> 8, "+NINE" -> 9, "+URZERO" -> ۰, "+URONE" -> ۱, "+URTWO" -
> ۲, "+URTHREE" -> ۳, "+URFOUR" -> ۴, "+URFIVE" -> ۵, "+URSIX" -> ۶,
"+URSEVEN" -> ۷, "+UREIGHT" -> ۸, "+URNINE" -> ۹, "+URQMARK" -> ؟,
"+URSTBD" -> ۔, "+URCOMMA" -> ،, "+URPERCENT" -> ٪, "+URSEMICOLON" -> ؛,
"+URSTAR" -> *, "+URDECIMAL" -> ٫, "+URTHSEP" -> ،];
!******************
read regex ["." -> ◌ٔ];
! ******************
!******************
! Consonants
!******************
! Simple Consonants
! Consonants Set 1
read regex [b -> ب, p -> پ, [t "_" d] -> ت, [t "`"] -> ٹ, x -> خ, [d
"_" d] -> د, [d "`"] -> ڈ, r -> ر, z -> ز, Z -> ژ, s -> س, S -> ش, G -
> غ, f -> ف, q -> ق, k -> ک, g -> گ, l -> ل, m -> م, n -> ن, v -> و, h
-> ہ, j -> ی, H -> ھ, [N 1] -> [ن گ], [N 2] -> [ن ج]];
! Consonants Set 2
read regex [[s 1] -> ث, [d "_" Z] -> ج, [t "_" S] -> چ, [h 1] -> ح, [z
1] -> ذ, [r "`"] -> ڑ, [S 1] -> ش, [s 2] -> ص, [z 2] -> ض, [t "_" d
1] -> ط, [z 3] -> ظ, [t "_" d 2] -> ۃ, [n "`"] -> ن, [l "`"] -> ل, [b
1] -> ب, [d "`" 1] -> ڈ, [g 1] -> گ, [r 1] -> ر];
! Consonants Set 3
read regex [[d "_" Z 1] -> ج, [r 1 1] -> ر];
! Other Aspirated Consonants
read regex [ ["_" h] -> ھ];
! Germinated Simple Consonants
read regex [[b "." b] -> [ب ّ], [p "." p] -> [پ ّ], [t "_" d "." t "_" d]
-> [ت ّ], [t "`" "." t "`"] -> [ٹ ّ], [s 1 "." s 1] -> [ث ّ], [d "_" Z "."
d "_" Z] -> [ج ّ], [t "_" S "." t "_" S] -> [چ ّ], [h 1 "." h 1] -> [ح ّ],
[x "." x] -> [خ ّ], [d "_" d "." d "_" d] -> [د ّ], [d "`" "." d "`"] ->
[ڈ ّ], [z 1 "." z 1] -> [ذ ّ], [r "." r] -> [ر ّ], [r "`" "." r "`"] -> [ڑ
```

```
ّ], [z "." z] -> [ّز], [Z "." Z] -> [ّژ], [s "." s] -> [ّس], [S "." S] ->
[ّش], [S 1 "." S 1] -> [ّش], [s 2 "." s 2] -> [ّص], [z 2 "." z 2] ->
[ّض], [t "_" d 1 "." t "_" d 1] -> [ّط], [z 3 "." z 3] -> [ّظ], [G "."
G] -> [ّغ], [f "." f] -> [ّف], [q "." q] -> [ّق], [k "." k] -> [ّک], [g
"." g] -> [ّگ], [l "." l] -> [ّل], [m "." m] -> [ّم], [n "." n] -> [ّن],
[v "." v] -> [ّو], [h "." h] -> [ّه], [j "." j] -> [ّی], [t "_" d 2 "." t
"_" d 2] -> [ّة], [n "`" "." n "`"] -> [ّن], [l "`" "." l "`"] -> [ّل],
[b 1 "." b 1] -> [ّب], [d "_" Z 1 "." d "_" Z 1] -> [ّج], [d "`" 1 "."
d "`" 1] -> [ّڈ], [g 1 "." g 1] -> [ّگ]];
! Germinated Aspirated Consonants
read regex [[b "." b "_" h] -> [ّبھ], [p "." p "_" h] -> [ّپھ], [t
"_" d "." t "_" d "_" h] -> [ّتھ], [t "`" "." t "`" "_" h] -> [ّٹھ],
[d "_" Z "." d "_" Z "_" h] -> [ّجھ], [t "_" S "." t "_" S "_" h] ->
[ّچھ], [d "_" d "." d "_" d "_" h] -> [ّدھ], [d "`" "." d "`" "_" h] -
> [ّڈھ], [r "." r "_" h] -> [ّرھ], [r "`" "." r "`" "_" h] -> [ّڑھ],
[k "." k "_" h] -> [ّکھ], [g "." g "_" h] -> [ّگھ], [l "." l "_" h] -
> [ّلھ], [m "." m "_" h] -> [ّمھ], [n "." n "_" h] -> [ّنھ], [v "." v
"_" h] -> [ّوھ]];
! *******************
! Nazalization
! *******************
read regex ["~" -> ن];
read regex [["~" 1] -> ن];
read regex ["~" -> ں || _ [.#. | " "]];
read regex [["~" 1] -> ں || _ [.#. | " "]];
!*******************
! Vowels
!*******************
! Vowels in the middle or at the end
read regex [A -> ا <- "@",آ, I -> [ِا], U -> [ُا], i -> [ِی], o -> [او], O
-> [َو], u -> [ُو], e -> [َی] <- ["}"],[ای]];
read regex [[e 3] -> [ع <- "?",[َے] <- [3 "}"],[اے];
! Vowel with diacritics
read regex [[A 1] -> ا, [I 1] -> ِ,[U 1] -> ُ, [i 1] -> [ِی], [o 1] ->
و, [O 1] -> [َو], [u 1] -> [ُو], [e 1] -> ی َ] <- [1 "}"],ی], [e 4] -> ,ے
ے َ] <- [4 "}"]]];
read regex [o -> و || [i 1] _ ["~" [.#. | " "]]];
read regex [["@" n] -> [ًا]];
read regex [[A 2] -> [ٰی], [A 3] -> ٰ, [I 2] -> [ء], [i 2] -> ٖ, [u 2]
-> ٗ, ["@" n 1] -> ً, [I 1 n 1] -> ٍ, [U 1 n 1] -> ٌ];
! *******************
! Special cases
! *******************
! Special cases of vowels
read regex [I -> ی || [VOWELS | "?"] _ [[h [ .#. | " " ]] | A | o |
"?"]];
read regex [[I 1] -> ی || CONSONANTS _ [[h [ .#. | " " ]] | A | o |
"?"]];
read regex [[U 1 v] -> [ّو] || CONSONANTS _ A];
read regex [[i 1 j] -> [ِّی] || CONSONANTS _ ];
read regex [[i 1] -> ی, [e 1] -> [ے َ] <- [1 "}"],[ے] || _ [.#. | " "]];
read regex [[I 2 i 1] -> [ءی], [I 2 e 1] -> [ءی], [I 2 "{" 1] -> [ء
ی], [I 2 o 1] -> [ءو], [I 2 u 1] -> [ءو], [I 2 U 1] -> [ءو], [I 2 O 1]
-> [ءو], [I 2 e 4] -> [ءے]];
```

```
! ******************
! Compound Words
read regex [["-" e "-"] -> [ِ " "], ["-" e 1 "-"] -> [ء " "]];
!******************
! Special words
read regex [[k I 1] -> [ہ ک] || _ [ .#. | " " ]];
read regex [[k I 1] -> [ہ ک] || [[.#. | " "] [[k j u 1 "~"] | [t "_" S
u 1 "~"] | [t "_" Z b] | [d "_" Z o 1] | [n A 1] | [b l]]] _ [.#. | "
"]];
read regex [ ["@" l "." l A 1 h] -> [ہ ّ ال]];
read regex [[k I 1] -> [ہ ک] || [ .#. | " " ] _ [ .#. | " " ]];
! ******************
! Multi-character symbols
read regex ["+DOT" -> "DOT", "+QMARK" -> "+QMARK", "+DECIMAL" ->
"+DECIMAL", "+COMMA" -> "+COMMA", "+NL" -> "+NL", "+THSEP" ->
"+THSEP", "+SEMICOLON" -> "+SEMICOLON", "+COLON" -> "+COLON",
"+PERCENT" -> "+PERCENT", "+ZERO" -> "+ZERO", "+ONE" -> "+ONE", "+TWO"
-> "+TWO", "+THREE" -> "+THREE", "+FOUR" -> "+FOUR", "+FIVE" ->
"+FIVE", "+SIX" -> "+SIX", "+SEVEN" -> "+SEVEN", "+EIGHT" -> "+EIGHT",
"+NINE" -> "+NINE", "+DSTBD" -> "+URSTBD", "+SSTBD" -> "+URSTBD",
"+GURDECIMAL" -> "+URDECIMAL", "+GURTHSEP" -> "+URTHSEP", "+GURZERO" -
> "+URZERO", "+GURONE" -> "+URONE", "+GURTWO" -> "+URTWO", "+GURTHREE"
-> "+URTHREE", "+GURFOUR" -> "+URFOUR", "+GURFIVE" -> "+URFIVE",
"+GURSIX" -> "+URSIX", "+GURSEVEN" -> "+URSEVEN", "+GUREIGHT" ->
"+UREIGHT", "+GURNINE" -> "+URNINE", "+URQMARK" -> "+URQMARK",
"+URSTBD" -> "+URSTBD", "+URCOMMA" -> "+URCOMMA", "+URPERCENT" ->
"+URPERCENT", "+URSEMICOLON" -> "+URSEMICOLON", "+URSTAR" ->
"+URSTAR", "+URDECIMAL" -> "+URDECIMAL", "+URTHSEP" -> "+URTHSEP",
"+URZERO" -> "+URZERO", "+URONE" -> "+URONE", "+URTWO" -> "+URTWO",
"+URTHREE" -> "+URTHREE", "+URFOUR" -> "+URFOUR", "+URFIVE" ->
"+URFIVE", "+URSIX" -> "+URSIX", "+URSEVEN" -> "+URSEVEN", "+UREIGHT"
-> "+UREIGHT", "+URNINE" -> "+URNINE", "+HIABRIVATION" -> "+URSTBD",
"+HIDECIMAL" -> "+URDECIMAL", "+HITHSEP" -> "+URTHSEP", "+HIZERO" ->
"+URZERO", "+HIONE" -> "+URONE", "+HITWO" -> "+URTWO", "+HITHREE" ->
"+URTHREE", "+HIFOUR" -> "+URFOUR", "+HIFIVE" -> "+URFIVE", "+HISIX" -
> "+URSIX", "+HISEVEN" -> "+URSEVEN", "+HIEIGHT" -> "+UREIGHT",
"+HININE" -> "+URNINE"];
compose net
```

Punjabi/Shahmukhi to UIT Finite-state Transducer

```
clear stack
set char-encoding UTF-8
!******************
!Definition of Varibales
!******************
! CONSONANTS | ASPICONSONANTS | ASPISHADCONSONANTS | ASPIZABARCONSO-
NANTS | ASPIZERCONSONANTS | ASPIPESHCONSONANTS | ASPISHADZABARCONSO-
NANTS | ASPISHADZERCONSONANTS | ASPISHADPESHCONSONANTS
! Simple Consonants
define                                                    CONSONANTS
[ب|پ|ت|ٹ|ث|ج|چ|ح|خ|د|ڈ|ذ|ر|ڑ|ز|ژ|س|ش|ص|ض|ط|ظ|ع|غ|ف|ق|ک|گ|ل|م|ن|ں|ہ|ھ|ا|آ|و|ی];
! Aspirated Consonants
define ASPICONSONANTS [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن|و] ھ];
! Aspirated Consonants with Shad
define ASPISHADCONSONANTS [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن|و] ّ ھ];
! Aspirated Consonants with ZABAR
define ASPIZABARCONSONANTS [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن|و] َ ھ];
```

```
! Aspirated Consonants with ZER
define ASPIZERCONSONANTS [[ھ ِ[و|ن|م|ل|گ|ک|ڑ|ر|ڈ|د|چ|ج|ٹ|ت|پ|ب];
! Aspiracted Consonants with PESH
define ASPIPESHCONSONANTS [[ھ ُ[و|ن|م|ل|گ|ک|ڑ|ر|ڈ|د|چ|ج|ٹ|ت|پ|ب];
! Aspirated Consonants with SHAD + ZABAR
! [[ھ َ ّ[ن|م|ل|گ|ک|ڑ|ر|ڈ|د|چ|ج|ٹ|ت|پ|ب]] | [ھ َ ّ[ن|م|ل|گ|ک|ڑ|ر|ڈ|د|چ|ج|ٹ|ت|پ|ب]
define ASPISHADZABARCONSONANTS [[ ّ[و|ن|م|ل|گ|ک|ڑ|ر|ڈ|د|چ|ج|ٹ|ت|پ|ب
ھ َ];
! Aspirated Consonants with SHAD + ZER
![[ھ ِ ّ[ن|م|ل|گ|ک|ڑ|ر|ڈ|د|چ|ج|ٹ|ت|پ|ب]] | [ھ ِ ّ[ن|م|ل|گ|ک|ڑ|ر|ڈ|د|چ|ج|ٹ|ت|پ|ب]
define ASPISHADZERCONSONANTS [[ ِ ّ[و|ن|م|ل|گ|ک|ڑ|ر|ڈ|د|چ|ج|ٹ|ت|پ|ب
ھ];
! Aspiracted Consonants with SHAD + PESH
define ASPISHADPESHCONSONANTS [[ ُ ّ[و|ن|م|ل|گ|ک|ڑ|ر|ڈ|د|چ|ج|ٹ|ت|پ|ب
ھ];
! English Numbers
define ENGNUMBER [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9];
! Urdu Numbers
define URDNUMBER [۰ | ۱ | ۲ | ۳ | ۴ | ۵ | ۶ | ۷ | ۸ | ۹];
define ENGNUMBEREXPRESSION [ [ENGNUMBER [ENGNUMBER]*] ];
define URNUMBEREXPRESSION [ [URDNUMBER [URDNUMBER]*] ];
!*******************
! Diacritics
!*******************
! Shahmukhi Diacritics ٰ ٚ ٛ ٖ ٗ ٘ ٙ ٚ ٛ ٜ ٝ ٞ ٟ ٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩
! here we will ignore Mada Sign, Hamza Above, Hamza Below, and Noong-
huna Mark diacritics
read regex [ َ -> 0, ِ -> [I 1], ُ -> [U 1], ٰ -> [A 3], ٖ -> [i 2], ٗ -> [u 2], ً -> ["@" n 1], ٍ -> [I 1 n 1], ٌ,"." <- ْ -> [U 1 n 1], ,0 <- ٚ,0 <- ٚ 0 <- ٔ,0 <- ٕ];
!*******************
! Consonants
!*******************
! Simple Consonants
! 1st Hamza is Hamze+Chotiyeh Ligature and 2nd Hamza is hamza itslef
read regex [ ب -> b, پ -> p, ت -> [t "_" d], ٹ -> [t "`"], ث -> [s 1], ج -> [d "_" Z], چ -> [t "_" S], ح -> [h 1], خ -> [x], د -> [d "_" d], ڈ -> [d "`"], ذ -> [z 1], ر -> r, ڑ -> [r "`"], ز -> z, ژ -> Z, س -> s, ش -> S, ص -> [s 2], ض -> [z 2], ط -> [t "_" d 1], ظ -> [z 3], ف -> f, ق -> q, ک -> k, گ -> g, ل -> l, م -> m, و -> v, ہ -> h, ء -> [I 2], ئ -> [I 2], ی -> j, غ -> G, ۃ -> [t "_" d 2], ڻ -> [n "`"], ھ -> h];
! Simple Consonants + SHAD
read regex [ [ ّب] -> [b "." b], [ ّپ] -> [p "." p], [ ّت] -> [t "_" d "." t "_" d], [ ّٹ] -> [t "`" "." t "`"], [ ّث] -> [s 1 "." s 1], [ ّج] -> [d "_" Z "." d "_" Z], [ ّچ] -> [t "_" S "." t "_" S], [ ّح] -> [h 1 "." h 1], [ ّخ] -> [x "." x], [ ّد] -> [d "_" d "." d "_" d], [ ّڈ] -> [d "`" "." d "`"], [ ّذ] -> [z 1 "." z 1], [ ّر] -> [r "." r], [ ّڑ] -> [r "`" "." r "`"], [ ّز] -> [z "." z], [ ّژ] -> [Z "." Z], [ ّس] -> [s "." s], [ ّش] -> [S "." S], [ ّص] -> [s 2 "." s 2], [ ّض] -> [z 2 "." z 2], [ ّط] -> [t "_" d 1 "." t "_" d 1], [ ّظ] -> [z 3 "." z 3], [ ّف] -> [f "." f], [ ّق] -> [q "." q], [ ّک] -> [k "." k], [ ّگ] -> [g "." g], [ ّل] -> [l "." l], [ ّم] -> [m "." m], [ ّہ] -> [h "." h], [ ّغ] -> [G "." G], [ ّڻ] -> [n "`" "." n "`"], [ ّھ] -> [h "." h], [ ّۃ] -> [t "_" d 2 "." t "_" d 2]];
```

```
! Aspirated Consonants
read regex [ ھ -> ["_" h] || [م|ل|گ|ک|ژ|ر|ڈ|د|چ|ج|ٹ|ت|پ|ب | v] _
];
! Aspirated Consonants + ZABAR
read regex [ [ب َ ھ] -> [b "_" h], [پ َ ھ] -> [p "_" h], [ت َ ھ] -> [t "_"
d "_" h], [ٹ َ ھ] -> [t "`" "_" h], [ج َ ھ] -> [d "_" Z "_" h], [چ َ ھ] ->
[t "_" S "_" h], [د َ ھ] -> [d "_" d "_" h], [ڈ َ ھ] -> [d "`" "_" h], [ر َ
ھ] -> [r "_" h], [ژ َ ھ] -> [r "`" "_" h], [ک َ ھ] -> [k "_" h], [گ َ ھ] ->
[g "_" h], [ل َ ھ] -> [l "_" h], [م َ ھ] -> [m "_" h]];
! Aspirated Consonants + ZER
read regex [ [ب ِ ھ] -> [b "_" h I 1], [پ ِ ھ] -> [p "_" h I 1], [ت ِ ھ] ->
[t "_" d "_" h I 1], [ٹ ِ ھ] -> [t "`" "_" h I 1], [ج ِ ھ] -> [d "_" Z "_"
h I 1], [چ ِ ھ] -> [t "_" S "_" h I 1], [د ِ ھ] -> [d "_" d "_" h I 1], [ڈ
ِ ھ] -> [d "`" "_" h I 1], [ر ِ ھ] -> [r "_" h I 1], [ژ ِ ھ] -> [r "`" "_"
h I 1], [ک ِ ھ] -> [k "_" h I 1], [گ ِ ھ] -> [g "_" h I 1], [ل ِ ھ] -> [l
"_" h I 1], [م ِ ھ] -> [m "_" h I 1]];
! Aspirated Consonants + ZER to ignore ZER while transliteration when
followed by CHOTI_YEH
read regex [ [ب ِ ھ] -> [b "_" h], [پ ِ ھ] -> [p "_" h], [ت ِ ھ] -> [t "_"
d "_" h], [ٹ ِ ھ] -> [t "`" "_" h], [ج ِ ھ] -> [d "_" Z "_" h], [چ ِ ھ] ->
[t "_" S "_" h], [د ِ ھ] -> [d "_" d "_" h], [ڈ ِ ھ] -> [d "`" "_" h], [ر ِ
ھ] -> [r "_" h], [ژ ِ ھ] -> [r "`" "_" h], [ک ِ ھ] -> [k "_" h], [گ ِ ھ] -
> [g "_" h], [ل ِ ھ] -> [l "_" h], [م ِ ھ] -> [m "_" h] || _ i];
! Aspirated Consonants + PESH
read regex [ [ب ُ ھ] -> [b "_" h U 1], [پ ُ ھ] -> [p "_" h U 1], [ت ُ ھ] ->
[t "_" d "_" h U 1], [ٹ ُ ھ] -> [t "`" "_" h U 1], [ج ُ ھ] -> [d "_" Z "_"
h U 1], [چ ُ ھ] -> [t "_" S "_" h U 1], [د ُ ھ] -> [d "_" d "_" h U 1], [ڈ
ُ ھ] -> [d "`" "_" h U 1], [ر ُ ھ] -> [r "_" h U 1], [ژ ُ ھ] -> [r "`" "_"
h U 1], [ک ُ ھ] -> [k "_" h U 1], [گ ُ ھ] -> [g "_" h U 1], [ل ُ ھ] -> [l
"_" h U 1], [م ُ ھ] -> [m "_" h U 1]];
! Aspirated Consonants + PESH to ignore PESH while transliteration
when followed by WAW
read regex [ [ب ُ ھ] -> [b "_" h], [پ ُ ھ] -> [p "_" h], [ت ُ ھ] -> [t "_"
d "_" h], [ٹ ُ ھ] -> [t "`" "_" h], [ج ُ ھ] -> [d "_" Z "_" h], [چ ُ ھ] ->
[t "_" S "_" h], [د ُ ھ] -> [d "_" d "_" h], [ڈ ُ ھ] -> [d "`" "_" h], [ر ُ
ھ] -> [r "_" h], [ژ ُ ھ] -> [r "`" "_" h], [ک ُ ھ] -> [k "_" h], [گ ُ ھ] -
> [g "_" h], [ل ُ ھ] -> [l "_" h], [م ُ ھ] -> [m "_" h] || _ u];
! Aspirated Consonants + SHAD
read regex [ [ب ّ ھ] -> [b "." b "_" h], [پ ّ ھ] -> [p "." p "_" h], [ت ّ
ھ] -> [t "_" d "." t "_" d "_" h], [ٹ ّ ھ] -> [t "`" "." t "`" "_" h],
[ج ّ ھ] -> [d "_" Z "." d "_" Z "_" h], [چ ّ ھ] -> [t "_" S "." t "_" S
"_" h], [د ّ ھ] -> [d "_" d "." d "_" d "_" h], [ڈ ّ ھ] -> [d "`" "." d "`"
"_" h], [ر ّ ھ] -> [r "." r "_" h], [ژ ّ ھ] -> [r "`" "." r "`" "_" h], [ک
ّ ھ] -> [k "." k "_" h], [گ ّ ھ] -> [g "." g "_" h], [ل ّ ھ] -> [l "." l
"_" h], [م ّ ھ] -> [m "." m "_" h], [ّ ب ھ] -> [b "." b "_" h], [ّ پ ھ] ->
[p "." p "_" h], [ّ ت ھ] -> [t "_" d "." t "_" d "_" h], [ّ ٹ ھ] -> [t
"`" "." t "`" "_" h], [ّ ج ھ] -> [d "_" Z "." d "_" Z "_" h], [ّ چ ھ] ->
[t "_" S "." t "_" S "_" h], [ّ د ھ] -> [d "_" d "." d "_" d "_" h], [ّ ڈ ھ
] -> [d "`" "." d "`" "_" h], [ّ ر ھ] -> [r "." r "_" h], [ّ ژ ھ] -> [r
"`" "." r "`" "_" h], [ّ ک ھ] -> [k "." k "_" h], [ّ گ ھ] -> [g "." g
"_" h], [ّ ل ھ] -> [l "." l "_" h], [ّ م ھ] -> [m "." m "_" h]];
! Aspirated Consonants + SHAD + ZABAR
read regex [ [ب َ ّ ھ] -> [b "." b "_" h], [پ َ ّ ھ] -> [p "." p "_" h], [ت
َ ّ ھ] -> [t "_" d "." t "_" d "_" h], [ٹ َ ّ ھ] -> [t "`" "." t "`" "_"
```

```
h], [ھ ً ّ ج] -> [d "_" Z "." d "_" Z "_" h], [ھ ً ّ چ] -> [t "_" S "." t
"_" S "_" h], [ھ ً ّ د] -> [d "_" d "." d "_" d "_" h], [ھ ً ّ ڈ] -> [d "`"
"." d "`" "_" h], [ھ ً ّ ر] -> [r "." r "_" h], [ھ ً ّ ڑ] -> [r "`" "." r
"`" "_" h], [ھ ً ّ ک] -> [k "." k "_" h], [ھ ً ّ گ] -> [g "." g "_" h], [ ّ ل
ھ ً] -> [l "." l "_" h], [ھ ً ّ م] -> [m "." m "_" h]];
! Aspirated Consonants + SHAD + ZER
read regex [ [ھ ِ ّ ب] -> [b "." b "_" h I 1], [ھ ِ ّ پ] -> [p "." p "_" h I
1], [ھ ِ ّ ت] -> [t "_" d "." t "_" d "_" h I 1], [ھ ِ ّ ٹ] -> [t "`" "." t
"`" "_" h I 1], [ھ ِ ّ ج] -> [d "_" Z "." d "_" Z "_" h I 1], [ھ ِ ّ چ] ->
[t "_" S "." t "_" S "_" h I 1], [ھ ِ ّ د] -> [d "_" d "." d "_" d "_" h I
1], [ھ ِ ّ ڈ] -> [d "`" "." d "`" "_" h I 1], [ھ ِ ّ ر] -> [r "." r "_" h I
1], [ھ ِ ّ ڑ] -> [r "`" "." r "`" "_" h I 1], [ھ ِ ّ ک] -> [k "." k "_" h I
1], [ھ ِ ّ گ] -> [g "." g "_" h I 1], [ھ ِ ّ ل] -> [l "." l "_" h I 1], [ ِ ّ م
ھ] -> [m "." m "_" h I 1]];
! Aspirated Consonants + SHAD + ZER while transliteration when fol-
lowed by CHOTI_YEH
read regex [ [ھ ِ ّ ب] -> [b "." b "_" h], [ھ ِ ّ پ] -> [p "." p "_" h], [ ت
ھ ِ ّ] -> [t "_" d "." t "_" d "_" h], [ھ ِ ّ ٹ] -> [t "`" "." t "`" "_"
h], [ھ ِ ّ ج] -> [d "_" Z "." d "_" Z "_" h], [ھ ِ ّ چ] -> [t "_" S "." t
"_" S "_" h], [ھ ِ ّ د] -> [d "_" d "." d "_" d "_" h], [ھ ِ ّ ڈ] -> [d "`"
"." d "`" "_" h], [ھ ِ ّ ر] -> [r "." r "_" h], [ھ ِ ّ ڑ] -> [r "`" "." r
"`" "_" h], [ھ ِ ّ ک] -> [k "." k "_" h], [ھ ِ ّ گ] -> [g "." g "_" h], [ ّ ل
ھ ِ] -> [l "." l "_" h], [ھ ِ ّ م] -> [m "." m "_" h] || _ i];
! Aspirated Consonants + SHAD + PESH
read regex [ [ھ ُ ّ ب] -> [b "." b "_" h U 1], [ھ ُ ّ پ] -> [p "." p "_" h U
1], [ھ ُ ّ ت] -> [t "_" d "." t "_" d "_" h U 1], [ھ ُ ّ ٹ] -> [t "`" "." t
"`" "_" h U 1], [ھ ُ ّ ج] -> [d "_" Z "." d "_" Z "_" h U 1], [ھ ُ ّ چ] ->
[t "_" S "." t "_" S "_" h U 1], [ھ ُ ّ د] -> [d "_" d "." d "_" d "_" h U
1], [ھ ُ ّ ڈ] -> [d "`" "." d "`" "_" h U 1], [ھ ُ ّ ر] -> [r "." r "_" h U
1], [ھ ُ ّ ڑ] -> [r "`" "." r "`" "_" h U 1], [ھ ُ ّ ک] -> [k "." k "_" h U
1], [ھ ُ ّ گ] -> [g "." g "_" h U 1], [ھ ُ ّ ل] -> [l "." l "_" h U 1], [ ُ ّ م
ھ] -> [m "." m "_" h U 1]];
! Aspirated Consonants + SHAD + PESH while transliteration when fol-
lowed by WAW
read regex [ [ھ ُ ّ ب] -> [b "." b "_" h], [ھ ُ ّ پ] -> [p "." p "_" h], [ ت
ھ ُ ّ] -> [t "_" d "." t "_" d "_" h], [ھ ُ ّ ٹ] -> [t "`" "." t "`" "_"
h], [ھ ُ ّ ج] -> [d "_" Z "." d "_" Z "_" h], [ھ ُ ّ چ] -> [t "_" S "." t
"_" S "_" h], [ھ ُ ّ د] -> [d "_" d "." d "_" d "_" h], [ھ ُ ّ ڈ] -> [d "`"
"." d "`" "_" h], [ھ ُ ّ ر] -> [r "." r "_" h], [ھ ُ ّ ڑ] -> [r "`" "." r
"`" "_" h], [ھ ُ ّ ک] -> [k "." k "_" h], [ھ ُ ّ گ] -> [g "." g "_" h], [ ّ ل
ھ ُ] -> [l "." l "_" h], [ھ ُ ّ م] -> [m "." m "_" h] || _ u];
!*******************
! Gol Heh at the end of a word
! Gol Heh is considered as a vowel A1 when it is at the end of a word
and is preceeded by a consonant. we will not convert it into A1, as
when we need to convert the text back in Shahmukhi, it will produce
ambiguity. we will handle this issue when we will convert UIT into
Gurmukhi. Two Gol Hehs at the end of word are considered as consonant
'h'. Or a Gol heh is considered as consonant 'h' when it is at the end
of word and is preceeded by a vowel.
! *******************
! Short Vowels with consonants
! WAW as vowel o after a consonant
read regex [و -> [o 1] || [[[CONSONANTS | ASPICONSONANTS] ّ] | CONSO-
NANTS | ASPICONSONANTS | ASPISHADCONSONANTS | j | v | n] _ ];
```

```
! ZABAR + WAW as au after a consonant
read regex [[ َ و] -> [O 1] || [[[CONSONANTS | ASPICONSONANTS] ّ] | CON-
SONANTS | ASPICONSONANTS | ASPISHADCONSONANTS | j | v | n] _ ];
! PESH + WAW as u after a consonant
read regex [[ ُ و] -> [u 1] || [[[CONSONANTS | ASPICONSONANTS] ّ] | CON-
SONANTS | ASPICONSONANTS | ASPISHADCONSONANTS | j | v | n] _ ];
! WAW as au after a consonant
read regex [و -> [O 1] || [ASPIZABARCONSONANTS | ASPISHADZABARCONSO-
NANTS] _ ];
! WAW as u after a consonant
read regex [و -> [u 1] || [ASPIPESHCONSONANTS | ASPISHADPESHCONSONANTS]
_ ];
! YEH as vowel e after a consonant
read regex [ ی -> [e 1] || [[[CONSONANTS | ASPICONSONANTS] ّ] | CONSO-
NANTS | ASPICONSONANTS | ASPISHADCONSONANTS | j | v | n] _ ];
! [[CONSONANTS | ASPICONSONANTS] ّ]
! ZABAR + YEH as vowel ai after a consonant
read regex [[ َ 1 "}"] <- [ی] || [[[CONSONANTS | ASPICONSONANTS] ّ] | CONSO-
NANTS | ASPICONSONANTS | ASPISHADCONSONANTS | j | v | n] _ ];
! ZER + YEH as vowel i after a consonant
read regex [[ ِ ی] -> [i 1] || [[[CONSONANTS | ASPICONSONANTS] ّ] | CON-
SONANTS | ASPICONSONANTS | ASPISHADCONSONANTS | j | v | n] _ ];
! YEH as i after a consonant
read regex [ی -> [i 1] || [ASPIZERCONSONANTS | ASPISHADZERCONSONANTS]
_ ];
! YEH as i after a consonant
read regex [1 "}"] <- ی] || [ASPIZABARCONSONANTS | ASPISHADZABARCONSO-
NANTS] _ ];
!*******************
! Vowels
!*******************
! Long Vowels not at the start of a word
! *******************
! Cases of Bari YEH
! Bari Yeh not at the end of a word
read regex [ے -> [e 4]];
! Bari YEH after ZABAR and not at the end of a word
read regex [ َ 4 "}"] <- ے]];
read regex [[ےء] -> [I 2 e 4], [ےئ] -> [I 2 e 4]];
read regex [4 "}"] <- ے] || [ASPIZABARCONSONANTS | ASPISHADZABARCONSO-
NANTS] _ ];
! Bari yeh will always form the vowel with a number 4 in the middle or
at the end of a word
! Vowels with AIN any where else
read regex [ ع -> ["?"] ];
read regex [ [ َع] -> ["?"] ];
read regex [ [ ِع] -> ["?" I 1], [ ُع] -> ["?" U 1]];
read regex [ [ع ی] -> ["?" e 1], [ع ِی] -> ["?" i 1], [ ,[1 "}" "?"] <- [ع َی
ع و]] -> ["?" o 1], [ ع َو] -> ["?" O 1], [ع ُو] -> ["?" u 1], [ع ے] ->
["?" e 4], [4 "}" "?"] <- [ع َے]];
! Vowels with ALEF any where else
read regex [ آ -> A];
read regex [ ا -> [A 1] ];
read regex [ [ َا] -> [A 1] ];
read regex [ [ ِا] -> I, [ ُا] -> U];
read regex [ [ا ی] -> e, [ا ِی] -> i, [او],"}" <- [ا َی] -> o, [ ا َو] -> O, [ا
ُو] -> u, [ا ے] -> [e 3], [3 "}"] <- [ا َے]];
```

```
! Vowels with AIN at the start of a word or a syllable
read regex [ ْ|.#.] || "?" <- ع | " "] _ ];
read regex [ [ْ|.#.] || "?" <- [َع | " "] _ ];
read regex [ [ِع] -> ["?" I 1], [ُع] -> ["?" U 1] || [.#. | ْ | " "] _
];
read regex [ [عى] -> ["?" e 1], [عِى] -> ["?" i 1], [,[1 "}" "?"] <- [عَى
عو]] -> ["?" o 1], [ عَو] -> ["?" O 1], [عُو] -> ["?" u 1], [عے] ->
["?" e 4], [ْ|.#.] || [4 "}" "?"] <- [عَے | " "] _ ];
! Vowels with ALEF at the start of a word or a syllable
read regex [ آ -> A || [.#. | ْ | " "] _ ];
read regex [ ْ|.#.] || ["@"] <- ا | " "] _ ];
read regex [ [ْ|.#.] || ["@"] <- [اَ | " "] _ ];
read regex [ [ِا] -> I, [ُا] -> U || [.#. | ْ | " "] _ ];
read regex [ [اى] -> e, [اِى] -> i, [او],"}" <- [اَى] -> o, [ اَو] -> O, [ا
وُ] -> u, [اے] -> [e 3], [ْ|.#.] || [3 "}"] <- [اَے | " "] _ ];
! ******************
! NOONGHUNA
read regex [ں -> "~"];
! ******************
! NOON when it is not considered as NOONGHUNA
! NOON as a nasalization
read regex [ن -> "~"];
! Noon followed by Noonghuna mark
read regex [[ّ ن] -> "~"];
! Noon + diacrictical marks
read regex [ن -> n || _ [َ |ٰ|ٗ|ٖ|ْ|ٍ|ٌ|ً|ُ|ِ]];
read regex [[ّن] -> [n "." n]];
! NOON followed by Heh Doachashmi as an aspirated consonant
read regex [[ن ھ] -> [n "_" h]];
! Noon + Diacritics+ Heh Doachashmi
read regex [[نَ ھ] -> [n "_" h َ]];
read regex [[نِ ھ] -> [n "_" h I 1]];
read regex [[نِ ھ] -> [n "_" h] || _ [i 1 .#.]];
read regex [[نِ ھ] -> [n "_" h ی_||[ِ];
read regex [[نُ ھ] -> [n "_" h U 1]];
read regex [[نُ ھ] -> [n "_" h و_||[ُ];
read regex [[نّ ھ] -> [n "." n "_" h]];
read regex [[نّ َھ] -> [n "." n "_" h َ]];
read regex [[نّ ِھ] -> [n "." n "_" h I 1]];
read regex [[نّ ِھ] -> [n "." n "_" h] || _ [i 1 .#.]];
read regex [[نّ ِھ] -> [n "." n "_" h ی_||[ِ];
read regex [[نّ ُھ] -> [n "." n "_" h U 1]];
read regex [[نّ ُھ] -> [n "." n "_" h و_||[ُ];
! Noon + Heh Doachashmi + Diacritics
read regex [[ن ھِ] -> [n "_" h] || _ [i 1 .#.]];
read regex [[ن ھّ] -> [n "." n "_" h]];
read regex [[ن ھّ ِ] -> [n "." n "_" h] || _ [i 1 .#.]];
! NOON before ALEF, ALEF Mada, Bari YEH, and diacritics
read regex [ن -> n || _ [ ن|ع|ے|آ|ا | i | ء|ئ|و|ى | v | j | I]];
! NOON at the end of a word
read regex [ن -> n || _ [ہ [.#. | " "]] ];
read regex [ن -> n || _ [.#. | ْ | " "] ];
! NOON at the start of a word
read regex [ن -> n || [.#. | ْ | " "] _ ];
```

```
!*******************
! Special Cases for Ain with Alef
!*******************
read regex [ع -> "?" || _ [i]];
read regex [[ع ا] -> [A 1 "?"] || ? _ ];
read regex [[ْ|.#.] || ["?" "@"] <- [ع ا | " "] _ ];
read regex [[ع ا] -> ["?" A 1]];
!*******************
! Special Cases for Waw
!*******************
read regex [و -> [o 1], ا -> [A 1], آ -> A, ع -> "?" || [I | 1 | 4] _ ];
read regex [و -> v, ی -> j || [ [? - [.#. | " "]] [آ|ا]] _ ];
!*******************
! Special Cases for Yeh
!*******************
! Yeh before Alef or Waw as I vowel
read regex [ی -> I || _ [ع|و|آ|ا]];
read regex [ی -> [I 1] || [ [[CONSONANTS | ASPICONSONANTS] ّ] | CONSO-
NANTS | ASPICONSONANTS | ASPISHADCONSONANTS | ASPIZABARCONSONANTS |
ASPIPESHCONSONANTS | ASPISHADZABARCONSONANTS | ASPISHADPESHCONSONANTS
| v | j] _ [ع|و|آ|ا]];
read regex [[ِ ی] -> [i 1] || [[[CONSONANTS | ASPICONSONANTS] ّ] | CON-
SONANTS | ASPICONSONANTS | ASPISHADCONSONANTS | ASPIZABARCONSONANTS |
ASPIPESHCONSONANTS | ASPISHADZABARCONSONANTS | ASPISHADPESHCONSONANTS
| v | j] _ [ع|و|آ|ا]];
read regex [[ی] -> [i 1] || [ASPIZERCONSONANTS | ASPISHADZERCONSO-
NANTS] _ [ع|و|آ|ا]];
! Yeh before Gol heh at the end of a word
read regex [ی -> I || _ [ه [.#. | " "]]];
read regex [ی -> [I 1] || [[[CONSONANTS | ASPICONSONANTS] ّ] | CONSO-
NANTS | ASPICONSONANTS | ASPISHADCONSONANTS | ASPIZABARCONSONANTS |
ASPIPESHCONSONANTS | ASPISHADZABARCONSONANTS | ASPISHADPESHCONSONANTS
| v | j] _ [ه [ .#. | " " ]]];
read regex [[ِ ی] -> [i 1] || [[[CONSONANTS | ASPICONSONANTS] ّ] | CON-
SONANTS | ASPICONSONANTS | ASPISHADCONSONANTS | ASPIZABARCONSONANTS |
ASPIPESHCONSONANTS | ASPISHADZABARCONSONANTS | ASPISHADPESHCONSONANTS
| v | j] _ [ه [ .#. | " " ]]];
read regex [[ی] -> [i 1] || [ASPIZERCONSONANTS | ASPISHADZERCONSO-
NANTS] _ [ه [ .#. | " " ]]];
! رباعیاں داعیوں
!***********read regex [ی -> [i 1] || [و|آ|ا] _ [ع [ی|ا]];
!*******************
! When Waw will be considered as a consonant
! Waw is followed by a vowel
read regex [و -> v || _ [ٰ|ٗ|ٖ|ِ|ُ|َ|ٍ|ٌ|ً|ے|ی|آ|ا | i | ه]];
read regex [و -> v || [و|ی|آ|ع|ا] _ [ی|و|آ|ع|ا]];
read regex [[ّو] -> [v "." v]];
! Waw + Diacritics+ Heh Doachashmi
read regex [[ه َو] -> [v "_" h َ]];
read regex [[ه ِو] -> [v "_" h I 1]];
read regex [[ه ِو] -> [v "_" h] || _ [i 1 .#.]];
read regex [[ه ِو] -> [v "_" h ی _ || [ِ];
read regex [[ه ُو] -> [v "_" h U 1]];
read regex [[ه ُو] -> [v "_" h و _ || [ُ];
read regex [[ه ّو] -> [v "." v "_" h]];
read regex [[ه َّو] -> [v "." v "_" h َ]];
```

```
read regex [[ھ ِ ّ و] -> [v "." v "_" h I 1]];
read regex [[ھ ِ ّ و] -> [v "." v "_" h] || _ [i 1 .#.]];
read regex [[ھ ِ ّ و] -> [v "." v "_" h ی _ ‖ [ِ]];
read regex [[ھ ُ ّ و] -> [v "." v "_" h U 1]];
read regex [[ھ ُ ّ و] -> [v "." v "_" h و _ ‖ [ُ]];
! Noon + Heh Doachashmi + Diacritics
read regex [[ھ ِ و] -> [v "_" h] || _ [i 1 .#.]];
read regex [[ھ ّ و] -> [v "." v "_" h]];
read regex [[ھ ّ ِ و] -> [v "." v "_" h] || _ [i 1 .#.]];
! Waw is followed by Shad
! چوّا، بوّا، سوّا
read regex [[ّ و] -> [U 1 v] || [CONSONANTS | ASPICONSONANTS | ASPISHAD-
CONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPISHADZABAR-
CONSONANTS | ASPISHADPESHCONSONANTS | j] _ [ا]];
! Waw at the start of a word
read regex [و -> v || [.#. | ْ | " "] _ ];
!*******************
! When Yeh will be considered as a consonant
read regex [[ّ ی] -> [j "." j]];
read regex [[ّ ی] -> [i 1 j] || [CONSONANTS | ASPICONSONANTS | ASPI-
SHADCONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPISHAD-
ZABARCONSONANTS | ASPISHADPESHCONSONANTS] _ ];
read regex [[ّ ی] -> [i 1 j] || [ASPIZERCONSONANTS | ASPISHADZERCONSO-
NANTS] _ ];
read regex [[ِ ّ ی] -> [i 1 j] || [CONSONANTS | ASPICONSONANTS | ASPI-
SHADCONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPISHAD-
ZABARCONSONANTS | ASPISHADPESHCONSONANTS] _ ];
! Yeh in between of two Alefs
! ایام، عیال، عیاں
read regex [ی -> j || [[.#. | و | ا] _ [[ع | ا] ["" | ْ]]];
! Yeh followed by Shad after Alef or Ain
read regex [[ّ ی] -> [j "." j] || [ع | ا] _ ];
! اعیان
read regex [ی -> j || [[.#. | ا] _ [[ع ا] ["" | ْ]]];
! Yeh is followed by a diacritic
read regex [ی -> j || _ [ ّ | ِ | ُ | َ | ٍ | ٌ | ً]];
! Yeh at the start of a word
read regex [ی -> j || [.#. | ْ | " "] _ ];
! Yeh at the end of a word
read regex [ی -> [i 1] || _ [.#. | ْ | " "]];
!*******************
! Special Cases
!*******************
! Special Cases for Hamza Waw
!*******************
read regex [[ئو] -> [I 2 o 1] , [ءو] -> [I 2 o 1]];
! Hamza + Waw gives sound of long vowel 'u' when it comes 2nd last in
the word and the word is a verb. جائوں، جھلوائوں، جیئوں
! In case, the word is a noun, the Hamza + waw gives the sound of 'o'.
so solve this problem, we need POS tagging of words.
read regex [[ئو] -> [I 2 u 1] , [ءو] -> [I 2 u 1] || _ [ن|ں]];
! *******************
! HAMZA after YEH as "@" vowel sound
read regex [[ءی] -> [I 2 i 1], [ئی] -> [I 2 i 1]];
! It seems that Hamza + Waw is never followed by Choti-Yeh. only word
that exist is سائویں (I think that it is wrong spelling of ساویں)
```

```
! Special strings at the end of a word
read regex [ [ٰی] -> A 2, [ٰی] -> A 2];
! First Hamza is Hamza and second is Hamza + Yeh Ligature
read regex [ [ٰی] -> A 2, [ٰی] -> A 2, [ًا] -> ["@" n], [ءی] -> [I 2 i
1], [ئی] -> [I 2 i 1], [ءو] -> [I 2 o 1], [ئو] -> [I 2 o 1]  || _
[.#. |ْ| " " | "-"] ];
read regex [[ءیے] -> [I 2 i 1 e], [ئیے] -> [I 2 i 1 e] ||  _ [.#. |
ْ| " " | "-"]];
! ******************
! Compound Words
!read regex [ ِ -> ["-" e "-"], ء -> ["-" e 1 "-"] || [CONSONANTS | AS-
PICONSONANTS] _ " "];
read regex [ ِ -> ["-" e "-"] || [CONSONANTS | ASPICONSONANTS] _ " "];
!******************
! Vowels alone
read regex [ْ|.#.] _ ["-"|""|ْ|.#.] || "@" <-ا| " " | "-"]];
read regex [[ِا],"@" <- [َا] -> I, [ُا] -> U || [.#. |ْ|.#.] _ ["-"|""|ْ| " " |
"-"]];
read regex [[ای] -> i, [ِای] -> i, [او],["}"] <- [َای] -> o, [َاو] -> O, [ُا
و] -> u, [اے] -> [e 3], [ْ|.#.] _ ["-"|""|ْ|.#.] || [3 "}"] <- [َاے| " " | "-"] ];
!******************
! Special words
read regex [[لله] -> ["@" l "." l A 1 h] || [? - ا] _ ];
read regex [[الله] -> ["@" l "." l A 1 h], [الله] -> ["@" l "." l A 1
h], [الّٰه] -> ["@" l "." l A 1 h], [الّه] -> ["@" l "." l A 1 h]];
read regex [[کہ] -> [k I 1], [وہ] -> [v h], [یہ] -> [j h] || [.#. | "
"] _ [.#. | " "]];
!******************
! Number
!******************
read regex [۰ -> "+URZERO", ۱ -> "+URONE", ۲ -> "+URTWO", ۳ ->
"+URTHREE", ۴ -> "+URFOUR", ۵ -> "+URFIVE", ۶ -> "+URSIX", ۷ ->
"+URSEVEN", ۸ -> "+UREIGHT", ۹ -> "+URNINE"];
! Decimal and thousand separator
read regex [، -> ["+URDECIMAL"], ، -> "+URTHSEP"];
read regex ["+DOT" -> "+URDECIMAL", "+COMMA" -> "+URTHSEP" || URNUMBE-
REXPRESSION _ URNUMBEREXPRESSION];
read regex ["0" -> "+ZERO", 1 -> "+ONE", 2 -> "+TWO", 3 -> "+THREE", 4
-> "+FOUR", 5 -> "+FIVE", 6 -> "+SIX", 7 -> "+SEVEN", 8 -> "+EIGHT", 9
-> "+NINE"];
read regex ["+DOT" -> "+DECIMAL", "+COMMA" -> "+THSEP" || ENGNUMBEREX-
PRESSION _ ENGNUMBEREXPRESSION];
read regex ["." -> "+DOT", ؟ -> "+URQMARK", ۔ -> "+URSTBD", ،->
"+URCOMMA", "," -> "+COMMA", "+NL" -> "+NL", ٪-> "+URPERCENT", ؛ ->
"+URSEMICOLON", * -> "+URSTAR"];
! Punjabi-Shahmukhi Text Normalization
read regex [0 <- "",گ <- گ̣,ڈ <- ڈ̣,ج <- ج̣,ب <- ب̣,ٹ <- ٹ];
compose net
```

UIT to Punjabi/Shahmukhi Finite-state Transducer

```
clear stack
set char-encoding UTF-8
!******************
!Definition of Varibales
```

```
!*******************
define CONSONANTS [b | [b "_" h] | p | [p "_" h] | [t "_" d] | [t "_"
d "_" h] | [t "`"] | [t "`" "_" h] | [s 1] | [d "_" Z] | [d "_" Z "_"
h] | [t "_" S] | [t "_" S "_" h] | [h 1] | x | [d "_" d] | [d "_" d
"_" h] | [d "`"] | [d "`" "_" h] | [z 1] | r | [r "_" h] | [r "`"] |
[r "`" "_" h] | z | Z| s | S | [s 2] | [z 2] | [t "_" d 1] | [z 3] | G
| f | q | k | [k "_" h] | g | [g "_" h] | l | [l "_" h] | m | [m "_"
h] | n | [n "_" h] | v | [v "_" h] | h | j | [t "_" d 2] | H | [N 1] |
[N 2] | [n "`"] | [l "`"] | [b 1] | [d "_" Z 1] | [d "`" 1] | [g 1]];
define VOWELS ["@" | A | I | U | e | "{" | i | u | O | o | [A 1] | [A
2] | [A 3] | [I 1] | [I 2] | [U 1] | [e 1] | ["{" 1] | [i 1] | [u 1] |
[O 1] | [o 1] | [e 3] | [e 4] | ["{" 3] | ["{" 4] | [i 2] | [u 2]];
!*******************
! Multi-character symbols
!*******************
read regex ["+DOT" -> ".", "+QMARK" -> "?", "+DECIMAL" -> ".",
"+COMMA" -> ",", "+THSEP" -> ",", "+SEMICOLON" -> ";", "+COLON" ->
":", "+PERCENT" -> ٪, "+ZERO" -> "0", "+ONE" -> 1, "+TWO" -> 2,
"+THREE" -> 3, "+FOUR" -> 4, "+FIVE" -> 5, "+SIX" -> 6, "+SEVEN" -> 7,
"+EIGHT" -> 8, "+NINE" -> 9, "+URZERO" -> ۰, "+URONE" -> ۱, "+URTWO" -
> ۲, "+URTHREE" -> ۳, "+URFOUR" -> ۴, "+URFIVE" -> ۵, "+URSIX" -> ۶,
"+URSEVEN" -> ۷, "+UREIGHT" -> ۸, "+URNINE" -> ۹, "+URQMARK" -> ؟,
"+URSTBD" -> -, "+URCOMMA" -> ،, "+URPERCENT" -> ٪, "+URSEMICOLON" -> ؛,
"+URSTAR" -> *, "+URDECIMAL" -> ٫, "+URTHSEP" -> ،];
!*******************
! Diacritics
!*******************
read regex ["." -> ْ];
!*******************
! Consonants
!*******************
! Simple Consonants
! Consonants Set 1
read regex [b -> ب, p -> پ, [t "_" d] -> ت, [t "`"] -> ٹ, x  -> خ, [d
"_" d] -> د, [d "`"] -> ڈ, r -> ر, z -> ز, Z -> ژ, s -> س, S -> ش, G -
> غ, f -> ف, q -> ق, k -> ک, g -> گ, l -> ل, m -> م, n -> ن, v -> و, h
-> ہ, j -> ی, H -> ھ, [N 1] -> [نگ], [N 2] -> [نج]];
! Consonants Set 2
read regex [[s 1] -> ث, [d "_" Z] -> ج, [t "_" S] -> چ, [h 1] -> ح, [z
1] -> ذ, [r "`"] -> ڑ, [S 1] -> ش, [s 2] -> ص, [z 2] -> ض, [t "_" d
1] -> ط, [z 3] -> ظ, [t "_" d 2] -> ة, [n "`"] -> □, [l "`"] -> ل, [b
1] -> ب, [d "`" 1] -> ڈ, [g 1] -> گ];
! Consonants Set 3
read regex [[d "_" Z 1] -> ج];
! Aspirated Consonants
read regex [ ["_" h] -> ھ];
! Germinated Simple Consonants
read regex [[b "." b] -> [بّ], [p "." p] -> [پّ], [t "_" d "." t "_" d]
-> [تّ], [t "`" "." t "`"] -> [ٹّ], [s 1 "." s 1] -> [ثّ], [d "_" Z "."
d "_" Z] -> [جّ], [t "_" S "." t "_" S] -> [چّ], [h 1 "." h 1] -> [حّ],
[x "." x] -> [خّ], [d "_" d "." d "_" d] -> [دّ], [d "`" "." d "`"] ->
[ڈّ], [z 1 "." z 1] -> [ذّ], [r "." r] -> [رّ], [r "`" "." r "`"] -> [ڑّ
], [z "." z] -> [زّ], [Z "." Z] -> [ژّ], [s "." s] -> [سّ], [S "." S] ->
[شّ], [S 1 "." S 1] -> [شّ], [s 2 "." s 2] -> [صّ], [z 2 "." z 2] ->
[ضّ], [t "_" d 1 "." t "_" d 1] -> [طّ], [z 3 "." z 3] -> [ظّ], [G "."
G] -> [غّ], [f "." f] -> [فّ], [q "." q] -> [قّ], [k "." k] -> [کّ], [g
"." g] -> [گّ], [l "." l] -> [لّ], [m "." m] -> [مّ], [n "." n] -> [نّ],
```

```
[v "." v] -> [ّو], [h "." h] -> [ّه], [j "." j] -> [ّی], [t "_" d 2 "." t
"_" d 2] -> [ّة], [n "`" "." n "`"] -> [ّ ], [l "`" "." l "`"] -> [ّل],
[b 1 "." b 1] -> [ّب], [d "_" Z 1 "." d "_" Z 1] -> [ّج], [d "`" 1 "."
d "`" 1] -> [ّڈ], [g 1 "." g 1] -> [ّگ]];
! Germinated Aspirated Consonants
read regex [[b "." b "_" h] -> [ّب ھ], [p "." p "_" h] -> [ّپ ھ], [t
"_" d "." t "_" d "_" h] -> [ّت ھ], [t "`" "." t "`" "_" h] -> [ّٹ ھ],
[d "_" Z "." d "_" Z "_" h] -> [ّج ھ], [t "_" S "." t "_" S "_" h] ->
[ّچ ھ], [d "_" d "." d "_" d "_" h] -> [ّد ھ], [d "`" "." d "`" "_" h] -
> [ّڈ ھ], [r "." r "_" h] -> [ّر ھ], [r "`" "." r "`" "_" h] -> [ّڑ ھ],
[k "." k "_" h] -> [ّک ھ], [g "." g "_" h] -> [ّگ ھ], [l "." l "_" h] -
> [ّل ھ], [m "." m "_" h] -> [ّم ھ], [n "." n "_" h] -> [ّن ھ], [v "." v
"_" h] -> [ّو ھ]];
!**********
! Nazalization
! **********
read regex ["~" -> ن];
read regex [["~" 1] -> ن];
read regex ["~" -> ں || _ [.#. | " "]];
read regex [["~" 1] -> ں || _ [.#. | " "]];
!**********
! Vowels
!**********
! Vowels
!**********
! Vowels in the middle or at the end
read regex [A -> ا <- "@",آ, I -> [ِا], U -> [ُا], i -> [ی ِا], o -> [وا], O
-> [وَا], u -> [وُا], e -> [یَا <- [ "}"] ,[یا]];
read regex [[e 3] -> [ع <- "?",[اَے <- [3 "}"],[اے];
! Vowel with diacritics
read regex [[A 1] -> ا, [I 1] -> ِ,[U 1] -> ُ, [i 1] -> [ی ِ], [o 1] ->
و, [O 1] -> [وَ], [u 1] -> [وُ], [e 1] -> یَ <- [1 "}"],ی], [e 4] -> ے,
ےَ <- [4 "}"]]];
! **********
! Special cases
! **********
! Special cases of vowels
read regex [o -> و || [i 1] _ ["~" [.#. | " "]]];
read regex [["@" n] -> [ًا]];
read regex [[A 2] -> [ٰی], [A 3] -> ٰ, [I 2] -> [ء], [i 2] -> ◌ِ, [u 2]
-> ◌ُ, ["@" n 1] -> ً, [I 1 n 1] -> ٍ, [U 1 n 1] -> ٌ];
read regex [I -> ی || [VOWELS | "?"] _ [[h [ .#. | " " ]] | A | o |
"?"]];
read regex [[I 1] -> ی || CONSONANTS _ [[h [ .#. | " " ]] | A | o |
"?"]];
read regex [[U 1 v] -> [ّو] || CONSONANTS _ A];
read regex [[i 1 j] -> [ِّی] || CONSONANTS _ ];
read regex [[i 1] -> ی, [e 1] -> [ےَ <- [1 "}"],[ے] || _ [.#. | " "]];
read regex [[I 2 i 1] -> [ی ء], [I 2 e 1] -> [ی ء], [I 2 "{" 1] -> [ء
ی], [I 2 o 1] -> [و ء], [I 2 u 1] -> [و ء], [I 2 U 1] -> [و ء], [I 2 O 1]
-> [وء], [I 2 e 4] -> [ے ء]];
read regex [[I 2 i 1 e] -> [ٰ|.#.] _ || [ے ی ء| " " | "-"] ];
! **********
! Compound Words
read regex [["-" e "-"] -> ِ, ["-" e 1 "-"] -> ء || _ [" "]];
```

```
!*******************
! Special words
read regex [[k I 1] -> [ہ ک] || [[.#. | " "] [[k j u 1 "~"] | [t "_" S
u 1 "~"] | [t "_" Z b] | [d "_" Z o 1] | [n A 1] | [b l]]] _ [.#. | "
"]];
read regex [ ["@" l "." l A 1 h] -> [ہ ّال]];
read regex [[k I 1] -> [ہ ک] || [ .#. | " " ] _ [ .#. | " " ]];
! *******************
! Multi-character symbols
read regex ["+DOT" -> "DOT", "+QMARK" -> "+QMARK", "+DECIMAL" ->
"+DECIMAL", "+COMMA" -> "+COMMA", "+NL" -> "+NL", "+THSEP" ->
"+THSEP", "+SEMICOLON" -> "+SEMICOLON", "+COLON" -> "+COLON",
"+PERCENT" -> "+PERCENT", "+ZERO" -> "+ZERO", "+ONE" -> "+ONE", "+TWO"
-> "+TWO", "+THREE" -> "+THREE", "+FOUR" -> "+FOUR", "+FIVE" ->
"+FIVE", "+SIX" -> "+SIX", "+SEVEN" -> "+SEVEN", "+EIGHT" -> "+EIGHT",
"+NINE" -> "+NINE", "+DSTBD" -> "+URSTBD", "+SSTBD" -> "+URSTBD",
"+GURDECIMAL" -> "+URDECIMAL", "+GURTHSEP" -> "+URTHSEP", "+GURZERO" -
> "+URZERO", "+GURONE" -> "+URONE", "+GURTWO" -> "+URTWO", "+GURTHREE"
-> "+URTHREE", "+GURFOUR" -> "+URFOUR", "+GURFIVE" -> "+URFIVE",
"+GURSIX" -> "+URSIX", "+GURSEVEN" -> "+URSEVEN", "+GUREIGHT" ->
"+UREIGHT", "+GURNINE" -> "+URNINE", "+URQMARK" -> "+URQMARK",
"+URSTBD" -> "+URSTBD", "+URCOMMA" -> "+URCOMMA", "+URPERCENT" ->
"+URPERCENT", "+URSEMICOLON" -> "+URSEMICOLON", "+URSTAR" ->
"+URSTAR", "+URDECIMAL" -> "+URDECIMAL", "+URTHSEP" -> "+URTHSEP",
"+URZERO" -> "+URZERO", "+URONE" -> "+URONE", "+URTWO" -> "+URTWO",
"+URTHREE" -> "+URTHREE", "+URFOUR" -> "+URFOUR", "+URFIVE" ->
"+URFIVE", "+URSIX" -> "+URSIX", "+URSEVEN" -> "+URSEVEN", "+UREIGHT"
-> "+UREIGHT", "+URNINE" -> "+URNINE", "+HIABRIVATION" -> "+URSTBD",
"+HIDECIMAL" -> "+URDECIMAL", "+HITHSEP" -> "+URTHSEP", "+HIZERO" ->
"+URZERO", "+HIONE" -> "+URONE", "+HITWO" -> "+URTWO", "+HITHREE" ->
"+URTHREE", "+HIFOUR" -> "+URFOUR", "+HIFIVE" -> "+URFIVE", "+HISIX" -
> "+URSIX", "+HISEVEN" -> "+URSEVEN", "+HIEIGHT" -> "+UREIGHT",
"+HININE" -> "+URNINE"];
compose net
```

Punjabi/Gurmukhi to UIT Finite-state Transducer

```
clear stack
set char-encoding UTF-8
!*******************
!Definition of Varibales
!*******************
!           Gurmukhi           Consonants,           total           38:
ਕਖਗਘਙਚਛਜਝਞਟਠਡਢਣਤਥਦਧਨਪਫਬਭਮਯਰਲਲ਼ਵਸ਼ਸਹਖ਼ਗ਼ਜ਼ੜਫ਼

define CONSONANTS [ਕ | ਖ | ਗ | ਘ | ਙ | ਚ | ਛ | ਜ | ਝ | ਞ | ਟ | ਠ | ਡ | ਢ |
ਣ | ਤ | ਥ | ਦ | ਧ | ਨ | ਪ | ਫ | ਬ | ਭ | ਮ | ਯ | ਰ | ਲ | ਲ਼ | ਵ | ਸ਼ | ਸ | ਹ | ਖ਼
| ਗ਼ | ਜ਼ | ੜ | ਫ਼];
define ENGNUMBER [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9];
define GURNUMBER [੦ | ੧ | ੨ | ੩ | ੪ | ੫ | ੬ | ੭ | ੮ | ੯];
define GURNUMBEREXPRESSION [ [GURNUMBER [GURNUMBER]*] ];
define ENGNUMBEREXPRESSION [ [ENGNUMBER [ENGNUMBER]*] ];
!*******************
! Rules for Gurmukhi to UIT Conversion
!*******************
!*******************
! Other Symbols
```

```
read regex [◌਼ -> ".", ◌ਃ -> H];
!*******************
! Rules for Consonants
!*******************
! Simple Consonants
read regex [ਕ -> k, ਖ -> [k "_" h], ਗ -> g, ਘ -> [g "_" h], ਙ -> [N 1], ਚ -
> [t "_" S], ਛ -> [t "_" S "_" h], ਜ -> [d "_" Z], ਝ -> [d "_" Z "_" h],
ਞ -> [N 2], ਟ -> [t "`"], ਠ -> [t "`" "_" h], ਡ -> [d "`"], ਢ -> [d "`"
"_" h], ਣ -> [n "`"], ਤ -> [t "_" d], ਥ -> [t "_" d "_" h], ਦ -> [d "_"
d], ਧ -> [d "_" d "_" h], ਨ -> [n], ਪ -> [p], ਫ -> [p "_" h], ਬ -> [b], ਭ -
> [b "_" h], ਮ -> [m], ਯ -> [j], ਰ -> [r], ਲ -> [l], ਲ਼ -> [l "`"], ਵ ->
[v], ਸ਼ -> [S], ਸ -> [s], ਹ -> [h], ਖ਼ -> [x], ਗ਼ -> [G], ਜ਼ -> [z], ੜ -> [r
"`"], ਫ਼ -> f];
read regex [[ਕ◌਼] -> q];
! Geminated Consonants
read regex [[◌ੱਕ] -> [k "." k], [◌ੱਖ] -> [k "." k "_" h], [◌ੱਗ] -> [g "."
g], [◌ੱਘ] -> [g "." g "_" h], [◌ੱਙ] -> [N 1 "." N 1], [◌ੱਚ] -> [t "_" S "."
t "_" S], [◌ੱਛ] -> [t "_" S "." t "_" S "_" h], [◌ੱਜ] -> [d "_" Z "." d
"_" Z], [◌ੱਝ] -> [d "_" Z "." d "_" Z "_" h], [◌ੱਞ] -> [N 2 "." N 2], [◌ੱ
ਟ] -> [t "`" "." t "`"], [◌ੱਠ] -> [t "`" "." t "`" "_" h], [◌ੱਡ] -> [d "`"
"." d "`"], [◌ੱਢ] -> [d "`" "." d "`" "_" h], [◌ੱਣ] -> [n "`" "." n "`"],
[◌ੱਤ] -> [t "_" d "." t "_" d], [◌ੱਥ] -> [t "_" d "." t "_" d "_" h], [◌ੱ
ਦ] -> [d "_" d "." d "_" d], [◌ੱਧ] -> [d "_" d "." d "_" d "_" h], [◌ੱਨ] -
> [n "." n], [◌ੱਪ] -> [p "." p], [◌ੱਫ] -> [p "." p "_" h], [◌ੱਬ] -> [b "."
b], [◌ੱਭ] -> [b "." b "_" h], [◌ੱਮ] -> [m "." m], [◌ੱਯ] -> [j "." j], [◌ੱਰ]
-> [r "." r], [◌ੱਲ] -> [l "." l], [◌ੱਲ਼] -> [l "`" "." l "`"], [◌ੱਵ] -> [v
"." v], [◌ੱਸ਼] -> [S "." S], [◌ੱਸ] -> [s "." s], [◌ੱਹ] -> [h "." h], [◌ੱਖ਼] ->
[x "." x], [◌ੱਗ਼] -> [G "." G], [◌ੱਜ਼] -> [z "." z], [◌ੱੜ] -> [r "`" "." r
"`"], [◌ੱਫ਼] -> [f "." f]];
read regex [[◌ੱਕ◌਼] -> [q "." q]];
! Aspirated Consonants
read regex [[◌਼ਹ] -> ["_" h] || [ਰ | ਲ | ਮ | ਨ | ਵ | ੜ] _ ];
! Special case for YA
read regex [ਯ -> [j h] || _ [.#. | " "]];
!*******************
! Rules for Nasalization
!*******************
! there are two main nasalization markers. ◌ਂ BINDI is used with vowel
characters and vowel symbols whose glyph goes above the tope horizon-
```

```
tal bar. otherwise, ੰ TIPI is used. But it is normal practice that
people use only BINDI.
read regex [ਂ -> "~", ੰ -> "~", ੱ -> ["~" 1]];
!*******************
! Rules for Vowels
!*******************
! Vowel signs
read regex [ਾ -> [A 1], ਿ -> [I 1], ੀ -> [i 1], ੁ -> [U 1], ੂ -> [u 1], ੇ ->
[e 1], ੈ -> ["{" 1], ੋ -> [o 1], ੌ -> [O 1]];
! Vowel characters not at the start of a word
read regex [ਅ -> "?", ਆ -> ["?" A 1], ਇ -> [I 2], ਈ -> [I 2 i 1], ਉ -> [I
2 U 1], ਊ -> [I 2 u 1], ਏ -> [I 2 e 1], ਐ -> "{", ਓ -> [I 2 o 1], ਔ -> O];
!read regex [[ਾ ਅ] -> "?"];
! when vowel sign U comes after a consonant and before some indepen-
dent vowel signs at the end of a word
read regex [[ੁ] -> [u 1] || CONSONANTS _ [[ਆ | ਈ | ਏ | ੰ | ਂ] [.#. | "
"]]];
read regex [ਆ -> [A 1], ਓ -> [o 1] || [ਇ | ਈ | ਓ | ਾ | ਿ | ੀ | ੁ | ੂ | ੇ | ੈ
| ੋ | ੌ | ੰ | ਂ | I] _ ];
read regex [ਇ -> I || [A | o | ਾ | ੋ | ਆ | ਓ] _ [ਆ | ਓ | ਾ | ੋ]];
! Vowel Characters at the start of a word
read regex [ਅ -> "@", ਆ -> A, ਇ -> I, ਈ -> i, ਉ -> U, ਊ -> u, ਏ -> e, ਐ ->
"{", ਓ -> o, ਔ -> O || [.#. | " "] _ ];
read regex [[ੁ] -> [u 1], [ਿ] -> [i 1] || CONSONANTS _ [.#. | " "] ];
!*******************
! Rules for specail strings
!*******************
! special words
read regex [[ਪ ੁ ਰ] -> [p u 1 r], [ਾ ਬ ਾ ਦ] -> [A b A 1 d "_" d], [ਅ ੱ ਲ ੍
ਹ ਾ] -> ["@" l "." l A 1 h] || [?] _ [.#. | " "]];
read regex [[ਅ ੱ ਲ ੍ ਹ ਾ] -> ["@" l "." l A 1 h], ੴ -> [O m], [ਨ] -> [n A
1], [ਯ ੇ] -> [j h], [ਵ ੇ] -> [v h], [ਵ ੈ] -> [v h] || [" " | .#.] _ [.#. |
" "]];
!*******************
! Rules for Numbers
!*******************
read regex [੦ -> "+GURZERO", ੧ -> "+GURONE", ੨ -> "+GURTWO", ੩ ->
"+GURTHREE", ੪ -> "+GURFOUR", ੫ -> "+GURFIVE", ੬ -> "+GURSIX", ੭ ->
"+GURSEVEN", ੮ -> "+GUREIGHT", ੯ -> "+GURNINE", ।  -> "+SSTBD", ॥ ->
"+DSTBD"];
! Decimal separator
read regex ["+DOT" -> "+GURDECIMAL", "+COMMA" -> "+GURTHSEP" || GUR-
NUMBEREXPRESSION _ GURNUMBEREXPRESSION];
read regex ["0" -> "+ZERO", 1 -> "+ONE", 2 -> "+TWO", 3 -> "+THREE", 4
-> "+FOUR", 5 -> "+FIVE", 6 -> "+SIX", 7 -> "+SEVEN", 8 -> "+EIGHT", 9
-> "+NINE"];
```

```
read regex ["+DOT" -> ["+DECIMAL"], "+COMMA" -> "+THSEP"  || ENGNUMBE-
REXPRESSION _ ENGNUMBEREXPRESSION];
read regex ["." -> "+DOT", "?" -> "+QMARK", "," -> "+COMMA", ";" ->
"+SEMICOLON", ":" -> "+COLON", "%" -> "+PERCENT", "+NL" -> "+NL"];
!*******************
! Rules for Text Normalization
!*******************
read regex [[ਇ ਿ] -> ਇ, [ਿ ਇ] -> ਇ, [ਇ ੀ] -> ਈ, [ਇ ੇ] -> ਏ, [ਉ ੁ] -> ਉ,
[ਉ ੂ] -> ਊ];
read regex [ੰ -> 0 || [? - ਕ] _ ];
read regex [[ਖ ੰ] -> ਖ਼, [ਗ ੰ] -> ਗ਼, [ਜ ੰ] -> ਜ਼, [ਫ ੰ] -> ਫ਼, [ਲ ੰ] -> ਲ਼, [ਸ ੰ]
-> ਸ਼];
compose net
```

UIT to Punjabi/Gurmukhi Finite-state Transducer

```
clear stack
set char-encoding UTF-8
!*******************
!Definition of Varibales
!*******************
define CONSONANTS [b | [b "_" h] | p | [p "_" h] | [t "_" d] | [t "_"
d "_" h] | [t "`"] | [t "`" "_" h] | [s 1] | [d "_" Z] | [d "_" Z "_"
h] | [t "_" S] | [t "_" S "_" h] | [h 1] | x | [d "_" d] | [d "_" d
"_" h] | [d "`"] | [d "`" "_" h] | [z 1] | r | [r "_" h] | [r "`"] |
[r "`" "_" h] | z | Z| s | S | [s 2] | [z 2] | [t "_" d 1] | [z 3] | G
| f | q | k | [k "_" h] | g | [g "_" h] | l | [l "_" h] | m | [m "_"
h] | n | [n "_" h] | v | [v "_" h] | h | j | [t "_" d 2] | H | [N 1] |
[N 2] | [n "`"] | [l "`"] | [b 1] | [d "_" Z 1] | [d "`" 1] | [g 1]];
define VOWELS ["@" | A | I | U | e | "{" | i | u | O | o | [A 1] | [A
2] | [A 3] | [I 1] | [I 2] | [U 1] | [e 1] | ["{" 1] | [i 1] | [u 1] |
[O 1] | [o 1] | [e 3] | [e 4] | ["{" 3] | ["{" 4] | [i 2] | [u 2]];
!*******************
! Multi-character symbols
!*******************
read regex ["+DOT" -> ".", "+QMARK" -> "?", "+DECIMAL" -> ".",
"+COMMA" -> ",", "+THSEP" -> ",", "+SEMICOLON" -> ";", "+COLON" ->
":", "+PERCENT" -> "%", "+ZERO" -> "0", "+ONE" -> 1, "+TWO" -> 2,
"+THREE" -> 3, "+FOUR" -> 4, "+FIVE" -> 5, "+SIX" -> 6, "+SEVEN" -> 7,
"+EIGHT" -> 8, "+NINE" -> 9, "+DSTBD" -> ॥, "+SSTBD" -> ।,
"+GURDECIMAL" -> ".", "+GURTHSEP" -> ",", "+GURTHSEP" -> ",",
"+GURZERO" -> ੦, "+GURONE" -> ੧, "+GURTWO" -> ੨, "+GURTHREE" -> ੩,
"+GURFOUR" -> ੪, "+GURFIVE" -> ੫, "+GURSIX" -> ੬, "+GURSEVEN" -> ੭,
"+GUREIGHT" -> ੮, "+GURNINE" -> ੯];
!*******************
! Diacritics
!*******************
read regex ["." -> ੰ];
!*******************
! Consonants
!*******************
! Simple Consonants
```

```
read regex [b -> ਬ, p -> ਪ, [t "_" d] -> ਤ, [t "`"] -> ਟ, x  -> ਖ਼, [d
"_" d] -> ਦ, [d "`"] -> ਡ, r -> ਰ, z -> ਜ਼, Z -> ਜ਼, s -> ਸ, S -> ਸ਼, G
-> ਗ਼, f -> ਫ਼, q -> [ਕ ◌ੁ], k -> ਕ, g -> ਗ, l -> ਲ, m -> ਮ, n -> ਨ, v -
> ਵ, h -> ਹ, j -> ਯ, H -> ◌ਃ, [N 1] -> ੜ, [N 2] -> ਢ];
! Consonants Set 2
read regex [[s 1] -> ਸ, [d "_" Z] -> ਜ, [t "_" S] -> ਚ, [h 1] -> ਹ,
[z 1] -> ਜ਼, [r "`"] -> ੜ, [S 1] -> ਸ਼, [s 2] -> ਸ, [z 2] -> ਜ਼, [t "_"
d 1] -> ਤ, [z 3] -> ਜ਼, [t "_" d 2] -> ਤ, [n "`"] -> ਣ, [l "`"] -> ਲ਼,
[b 1] -> ਬ, [d "`" 1] -> ਡ, [g 1] -> ਗ];
! Consonants Set 3
read regex [[d "_" Z 1] -> ਜ];
! Aspirated Consonants
read regex [ ["_" h] -> [◌ੁ ਹ]];
read regex [[b "_" h] -> ਭ, [p "_" h] -> ਫ, [t "_" d "_" h] -> ਥ, [t
"`" "_" h] -> ਠ, [d "_" Z "_" h] -> ਝ, [t "_" S "_" h] -> ਛ, [d "_" d
"_" h] -> ਧ, [d "`" "_" h] -> ਢ, [k "_" h] -> ਖ, [g "_" h] -> ਘ];
! Germinated Simple Consonants
read regex [[b "." b] -> [◌ੱ ਬ], [p "." p] -> [◌ੱ ਪ], [t "_" d "." t "_"
d] -> [◌ੱ ਤ], [t "`" "." t "`"] -> [◌ੱ ਟ], [s 1 "." s 1] -> [◌ੱ ਸ], [d "_"
Z "." d "_" Z] -> [◌ੱ ਜ], [t "_" S "." t "_" S] -> [◌ੱ ਚ], [h 1 "." h 1]
-> [◌ੱ ਹ], [x "." x] -> [◌ੱ ਖ਼], [d "_" d "." d "_" d] -> [◌ੱ ਦ], [d "`" "."
d "`"] -> [◌ੱ ਡ], [z 1 "." z 1] -> [◌ੱ ਜ਼], [r "." r] -> [◌ੱ ਰ], [r "`" "."
r "`"] -> [◌ੱ ੜ], [z "." z] -> [◌ੱ ਜ਼], [Z "." Z] -> [◌ੱ ਜ਼], [s "." s] -> [◌ੱ
ਸ], [S "." S] -> [◌ੱ ਸ਼], [S 1 "." S 1] -> [◌ੱ ਸ਼], [s 2 "." s 2] -> [◌ੱ ਸ],
[z 2 "." z 2] -> [◌ੱ ਜ਼], [t "_" d 1 "." t "_" d 1] -> [◌ੱ ਤ], [z 3 "." z
3] -> [◌ੱ ਜ਼], [G "." G] -> [◌ੱ ਗ਼], [f "." f] -> [◌ੱ ਫ਼], [q "." q] -> [◌ੱ
ਕ ◌ੁ], [k "." k] -> [◌ੱ ਕ], [g "." g] -> [◌ੱ ਗ], [l "." l] -> [◌ੱ ਲ], [m "."
m] -> [◌ੱ ਮ], [n "." n] -> [◌ੱ ਨ], [v "." v] -> [◌ੱ ਵ], [h "." h] -> [◌ੱ ਹ],
[j "." j] -> [◌ੱ ਯ], [t "_" d 2 "." t "_" d 2] -> [◌ੱ ਤ], [n "`" "." n
"`"] -> [◌ੱ ਣ], [l "`" "." l "`"] -> [◌ੱ ਲ਼], [b 1 "." b 1] -> [◌ੱ ਬ], [d
"_" Z 1 "." d "_" Z 1] -> [◌ੱ ਜ], [d "`" 1 "." d "`" 1] -> [◌ੱ ਡ], [g 1
"." g 1] -> [◌ੱ ਗ], [N 1 "." N 1] -> [◌ੱ ੜ], [N 2 "." N 2] -> [◌ੱ ਢ]];
! Germinated Aspirated Consonants
read regex [[b "." b "_" h] -> [◌ੱ ਭ], [p "." p "_" h] -> [◌ੱ ਫ], [t "_" d
"." t "_" d "_" h] -> [◌ੱ ਥ], [t "`" "." t "`" "_" h] -> [◌ੱ ਠ], [d "_" Z
"." d "_" Z "_" h] -> [◌ੱ ਝ], [t "_" S "." t "_" S "_" h] -> [◌ੱ ਛ], [d
```

```
"_" d "." d "_" d "_" h] -> [ੱਧ], [d "`" "." d "`" "_" h] -> [ੱਢ], [k
"." k "_" h] -> [ੱਖ], [g "." g "_" h] -> [ੱਘ]];
! HEH at the end of a word as vowel A
read regex [h -> ਾ || [CONSONANTS] _ [.#. | " "]];
! special cases for YA
!read regex [[j h] -> ਯ || _ .#.];
! ********************
! Nazalization
read regex ["~" -> ਂ];
read regex ["~" -> ੰ || [CONSONANTS | ੁ | ੂ] _ ];
read regex [ ["~" 1] -> ੰ];
!********************
! Vowels
!********************
! Vowel characters
read regex [A -> ਆ, "@" -> ਅ, I -> ਇ, U -> ਉ, i -> ਈ, o -> ਓ, O ->
ਔ, u -> ਊ, e -> ਏ, "{" -> ਐ];
read regex [[e 3] -> ਏ, ["{" 3] -> ਐ];
! Vowel signs
read regex [[A 1] -> ਾ, [I 1] -> ਿ, [U 1] -> ੁ, [i 1] -> ੀ, [o 1] -> ੋ,
[O 1] -> ੌ, [u 1] -> ੂ, [e 1] -> ੇ, ["{" 1] -> ੈ, [e 4] -> ੇ, ["{" 4] -
> ੈ];
! ********************
! Special cases
! ********************
! Special cases of vowels
read regex [["@" n] -> ਨ || [? - .#.] _ ];
read regex [[A 2] -> ਾ, [A 3] -> ਾ, [I 2] -> ਇ, [i 2] -> ੀ, [u 2] -
> ੂ, ["@" n 1] -> ਨ, [I 1 n 1] -> [ਿ ਨ], [U 1 n 1] -> [ੁ ਨ]];
read regex [[u 1] -> ੁ || CONSONANTS _ [[A | i | e] .#.]];
! Vowel with Ain
read regex ["?" -> ਅ];
read regex ["?" -> ਾ || CONSONANTS _ ];
read regex [["?" A 1] -> ਆ, A 1 "?" -> [ਾ ਇ]];
read regex [["?" I 1] -> ਇ, ["?" U 1] -> ਉ, ["?" e 1] -> ਏ, ["?" i 1]
-> ਈ, ["?" "{" 1] -> ਐ, ["?" o 1] -> ਓ, ["?" O 1] -> ਔ, ["?" u 1] ->
ਊ, ["?" e 4] -> ਏ, ["?" "{" 4] -> ਐ];
!read regex [["@" "?"] -> ਏ, ["?" I 1] -> ਇ, ["?" U 1] -> ਉ, ["?" e 1]
-> ਏ, ["?" i 1] -> ਈ, ["?" "{" 1] -> ਐ, ["?" o 1] -> ਓ, ["?" O 1] ->
ਔ, ["?" u 1] -> ਊ, ["?" e 4] -> ਏ, ["?" "{" 4] -> ਐ || [.#. | " " |
"."] _ ];
! Ain after the final i sound in the word
```

```
read regex [["?" i 1] -> [◌ਾ ਈ] || _ [.#. | " "]];
read regex [[A 1] -> ਆ, [I 1] -> ਇ, [U 1] -> ਉ, [i 1] -> ਈ, [o 1] -> ਓ, [O 1] -> ਔ, [u 1] -> ਊ, [e 1] -> ਏ, ["{" 1] -> ਐ, [e 4] -> ਏ, ["{" 4] -> ਐ || [VOWELS | ਈ | ਏ | ਐ | ਓ | ਊ | ਉ | ਔ] _ ];
read regex [[I 2 i 1] -> ਈ, [I 2 e 1] -> ਏ, [I 2 "{" 1] -> ਐ, [I 2 o 1] -> ਓ , [I 2 u 1] -> ਊ, [I 2 U 1] -> ਉ, [I 2 O 1] -> ਔ, [I 2 e 4] -> ਏ];
! ******************
! Compound Words
read regex [["-" e "-"] -> ["-" ਏ "-"], ["-" e 1 "-"] -> ["-" ਏ "-"] || _ [" "]];
!******************
! Special words
read regex [ ["@" l "." l A 1 h] -> [ਅ ◌ੱ ਲ ◌੍ ਹ ◌ਾ], [p u 1 r] -> [ਪ ◌ੁ ਰ], [h h] -> ਹ || _ [.#. | " "]];
read regex [["@" l "." l A 1 h] -> [ਅ ◌ੱ ਲ ◌੍ ਹ ◌ਾ], [v h] -> [ਵ ◌ੇ], [j h] -> [ਯ ◌ੇ], [n A 1] -> ਨ, [O m] -> ੴ || [.#. | " " ] _ [ .#. | " " ]];
! ******************
! Multi-character symbols
read regex ["+DOT" -> "DOT", "+QMARK" -> "+QMARK", "+DECIMAL" -> "+DECIMAL", "+COMMA" -> "+COMMA", "+NL" -> "+NL", "+THSEP" -> "+THSEP", "+SEMICOLON" -> "+SEMICOLON", "+COLON" -> "+COLON", "+PERCENT" -> "+PERCENT", "+ZERO" -> "+ZERO", "+ONE" -> "+ONE", "+TWO" -> "+TWO", "+THREE" -> "+THREE", "+FOUR" -> "+FOUR", "+FIVE" -> "+FIVE", "+SIX" -> "+SIX", "+SEVEN" -> "+SEVEN", "+EIGHT" -> "+EIGHT", "+NINE" -> "+NINE", "+DSTBD" -> "+DSTBD", "+SSTBD" -> "+SSTBD", "+GURDECIMAL" -> "+GURDECIMAL", "+GURTHSEP" -> "+GURTHSEP", "+GURZERO" -> "+GURZERO", "+GURONE" -> "+GURONE", "+GURTWO" -> "+GURTWO", "+GURTHREE" -> "+GURTHREE", "+GURFOUR" -> "+GURFOUR", "+GURFIVE" -> "+GURFIVE", "+GURSIX" -> "+GURSIX", "+GURSEVEN" -> "+GURSEVEN", "+GUREIGHT" -> "+GUREIGHT", "+GURNINE" -> "+GURNINE", "+URQMARK" -> "+QMARK", "+URSTBD" -> "+SSTBD", "+URCOMMA" -> "+COMMA", "+URPERCENT" -> "+PERCENT", "+URSEMICOLON" -> "+SEMICOLON", "+URSTAR" -> "*", "+URDECIMAL" -> "+GURDECIMAL", "+URTHSEP" -> "+GURTHSEP", "+URZERO" -> "+GURZERO", "+URONE" -> "+GURONE", "+URTWO" -> "+GURTWO", "+URTHREE" -> "+GURTHREE", "+URFOUR" -> "+GURFOUR", "+URFIVE" -> "+GURFIVE", "+URSIX" -> "+GURSIX", "+URSEVEN" -> "+GURSEVEN", "+UREIGHT" -> "+GUREIGHT", "+URNINE" -> "+GURNINE", "+HIABRIVATION" -> "+DOT", "+HIDECIMAL" -> "+GURDECIMAL", "+HITHSEP" -> "+GURTHSEP", "+HIZERO" -> "+GURZERO", "+HIONE" -> "+GURONE", "+HITWO" -> "+GURTWO", "+HITHREE" -> "+GURTHREE", "+HIFOUR" -> "+GURFOUR", "+HIFIVE" -> "+GURFIVE", "+HISIX" -> "+GURSIX", "+HISEVEN" -> "+GURSEVEN", "+HIEIGHT" -> "+GUREIGHT", "+HININE" -> "+GURNINE"];
!"+URSTBD" -> "+SSTBD", "+URQMARK" -> "+QMARK", "+URTHSEP" -> "+GURTHSEP", "+URDECIMAL" -> "+GURDECIMAL", "+URSEMICOLON" -> "+SEMICOLON", +URPERCENT" -> "+PERCENT", "+URCOMMA" -> "+COMMA", "+URSTAR" -> "*",
compose net
```

Seraiki/Shamukhi to UIT Finite-state Transducer

```
clear stack
set char-encoding UTF-8
!*******************
!Definition of Varibales
!*******************
! Simple Consonants
define                                                    CONSONANTS
[ب|پ|ت|ٹ|ث|ج|چ|ح|خ|د|ڈ|ذ|ر|ڑ|ز|س|ش|ص|ض|ط|ظ|ع|غ|ف|ق|ک|گ|ل|م|ن|ں|ہ|ۃ|و|ی|ے|ۓ|ڀ|ڄ|ڃ|ڇ|ڈ|ڌ|ڍ|ڏ|ڑ|ږ|ژ|ښ|ڙ|ڛ|ڜ|ڝ|ڞ|ڟ|ڠ|ڡ|ڢ|ڣ|ڤ|ڥ|ڦ|ڧ|ڨ|ڪ|ګ|ڬ|ڭ|ڮ|ڰ|ڱ|ڲ|ڳ|ڴ|ڵ|ڶ|ڷ|ڸ|ڹ|ڻ|ڼ|ڽ|ھ|ۀ|ہ|ۂ|ۃ|ۄ|ۅ|ۆ|ۇ|ۈ|ۉ|ۊ|ۋ|ی|ۍ|ێ|ې|ۑ|ے|ۓ|ە|ۮ|ۯ|ۺ|ۻ|ۼ|ۿ|ﭐ|ﭑ|ﭒ|ﭓ|ﭔ|ﭕ|ﭖ|ﭗ|ﭘ|ﭙ|ﭚ|ﭛ|ﭜ|ﭝ|ﭞ|ﭟ|ﭠ|ﭡ|ﭢ|ﭣ|ﭤ|ﭥ|ﭦ|ﭧ|ﭨ|ﭩ|ﭪ|ﭫ|ﭬ|ﭭ|ﭮ|ﭯ|ﭰ|ﭱ|ﭲ|ﭳ|ﭴ|ﭵ|ﭶ|ﭷ|ﭸ|ﭹ|ﭺ|ﭻ|ﭼ|ﭽ|ﭾ|ﭿ|ﮀ|ﮁ|ﮂ|ﮃ|ﮄ|ﮅ|ﮆ|ﮇ|ﮈ|ﮉ|ﮊ|ﮋ|ﮌ|ﮍ|ﮎ|ﮏ|ﮐ|ﮑ|ﮒ|ﮓ|ﮔ|ﮕ|ﮖ|ﮗ|ﮘ|ﮙ|ﮚ|ﮛ|ﮜ|ﮝ|ﮞ|ﮟ|ﮠ|ﮡ|ﮢ|ﮣ|ﮤ|ﮥ|ﮦ|ﮧ|ﮨ|ﮩ|ﮪ|ﮫ|ﮬ|ﮭ|ﮮ|ﮯ|ﮰ|ﮱ|ﯓ|ﯔ|ﯕ|ﯖ|ﯗ|ﯘ|ﯙ|ﯚ|ﯛ|ﯜ|ﯝ|ﯞ|ﯟ|ﯠ|ﯡ|ﯢ|ﯣ|ﯤ|ﯥ|ﯦ|ﯧ|ﯨ|ﯩ|ﯪ|ﯫ|ﯬ|ﯭ|ﯮ|ﯯ|ﯰ|ﯱ|ﯲ|ﯳ|ﯴ|ﯵ|ﯶ|ﯷ|ﯸ|ﯹ|ﯺ|ﯻ|ﯼ|ﯽ|ﯾ|ﯿ|ﰀ|ﰁ|ﰂ|ﰃ|ﰄ|ﰅ|ﰆ|ﰇ|ﰈ|ﰉ|ﰊ|ﰋ|ﰌ|ﰍ|ﰎ|ﰏ|ﰐ|ﰑ|ﰒ|ﰓ|ﰔ|ﰕ|ﰖ|ﰗ|ﰘ|ﰙ|ﰚ|ﰛ|ﰜ|ﰝ|ﰞ|ﰟ|ﰠ|ﰡ|ﰢ|ﰣ|ﰤ|ﰥ|ﰦ|ﰧ|ﰨ|ﰩ|ﰪ|ﰫ|ﰬ|ﰭ|ﰮ|ﰯ|ﰰ|ﰱ|ﰲ|ﰳ|ﰴ|ﰵ|ﰶ|ﰷ|ﰸ|ﰹ|ﰺ|ﰻ|ﰼ|ﰽ|ﰾ|ﰿ|ﱀ|ﱁ|ﱂ|ﱃ|ﱄ|ﱅ|ﱆ|ﱇ|ﱈ|ﱉ|ﱊ|ﱋ|ﱌ|ﱍ|ﱎ|ﱏ|ﱐ|ﱑ|ﱒ|ﱓ|ﱔ|ﱕ|ﱖ|ﱗ|ﱘ|ﱙ|ﱚ|ﱛ|ﱜ|ﱝ|ﱞ|ﱟ|ﱠ|ﱡ|ﱢ|ﱣ|ﱤ|ﱥ|ﱦ|ﱧ|ﱨ|ﱩ|ﱪ|ﱫ|ﱬ|ﱭ|ﱮ|ﱯ|ﱰ|ﱱ|ﱲ|ﱳ|ﱴ|ﱵ|ﱶ|ﱷ|ﱸ|ﱹ|ﱺ|ﱻ|ﱼ|ﱽ|ﱾ|ﱿ|ﲀ|ﲁ|ﲂ|ﲃ|ﲄ|ﲅ|ﲆ|ﲇ|ﲈ|ﲉ|ﲊ|ﲋ|ﲌ|ﲍ|ﲎ|ﲏ|ﲐ|ﲑ|ﲒ|ﲓ|ﲔ|ﲕ|ﲖ|ﲗ|ﲘ|ﲙ|ﲚ|ﲛ|ﲜ|ﲝ|ﲞ|ﲟ|ﲠ|ﲡ|ﲢ|ﲣ|ﲤ|ﲥ|ﲦ|ﲧ|ﲨ|ﲩ|ﲪ|ﲫ|ﲬ|ﲭ|ﲮ|ﲯ|ﲰ|ﲱ|ﲲ|ﲳ|ﲴ|ﲵ|ﲶ|ﲷ|ﲸ|ﲹ|ﲺ|ﲻ|ﲼ|ﲽ|ﲾ|ﲿ|ﳀ|ﳁ|ﳂ|ﳃ|ﳄ|ﳅ|ﳆ|ﳇ|ﳈ|ﳉ|ﳊ|ﳋ|ﳌ|ﳍ|ﳎ|ﳏ|ﳐ|ﳑ|ﳒ|ﳓ|ﳔ|ﳕ|ﳖ|ﳗ|ﳘ|ﳙ|ﳚ|ﳛ|ﳜ|ﳝ|ﳞ|ﳟ|ﳠ|ﳡ|ﳢ|ﳣ|ﳤ|ﳥ|ﳦ|ﳧ|ﳨ|ﳩ|ﳪ|ﳫ|ﳬ|ﳭ|ﳮ|ﳯ|ﳰ|ﳱ|ﳲ|ﳳ|ﳴ|ﳵ|ﳶ|ﳷ|ﳸ|ﳹ|ﳺ|ﳻ|ﳼ|ﳽ|ﳾ|ﳿ|ﴀ|ﴁ|ﴂ|ﴃ|ﴄ|ﴅ|ﴆ|ﴇ|ﴈ|ﴉ|ﴊ|ﴋ|ﴌ|ﴍ|ﴎ|ﴏ|ﴐ|ﴑ|ﴒ|ﴓ|ﴔ|ﴕ|ﴖ|ﴗ|ﴘ|ﴙ|ﴚ|ﴛ|ﴜ|ﴝ|ﴞ|ﴟ|ﴠ|ﴡ|ﴢ|ﴣ|ﴤ|ﴥ|ﴦ|ﴧ|ﴨ|ﴩ|ﴪ|ﴫ|ﴬ|ﴭ|ﴮ|ﴯ|ﴰ|ﴱ|ﴲ|ﴳ|ﴴ|ﴵ|ﴶ|ﴷ|ﴸ|ﴹ|ﴺ|ﴻ|ﴼ|ﴽ|﴾|﴿|ﵐ|ﵑ|ﵒ|ﵓ|ﵔ|ﵕ|ﵖ|ﵗ|ﵘ|ﵙ|ﵚ|ﵛ|ﵜ|ﵝ|ﵞ|ﵟ|ﵠ|ﵡ|ﵢ|ﵣ|ﵤ|ﵥ|ﵦ|ﵧ|ﵨ|ﵩ|ﵪ|ﵫ|ﵬ|ﵭ|ﵮ|ﵯ|ﵰ|ﵱ|ﵲ|ﵳ|ﵴ|ﵵ|ﵶ|ﵷ|ﵸ|ﵹ|ﵺ|ﵻ|ﵼ|ﵽ|ﵾ|ﵿ|ﶀ|ﶁ|ﶂ|ﶃ|ﶄ|ﶅ|ﶆ|ﶇ|ﶈ|ﶉ|ﶊ|ﶋ|ﶌ|ﶍ|ﶎ|ﶏ|ﶒ|ﶓ|ﶔ|ﶕ|ﶖ|ﶗ|ﶘ|ﶙ|ﶚ|ﶛ|ﶜ|ﶝ|ﶞ|ﶟ|ﶠ|ﶡ|ﶢ|ﶣ|ﶤ|ﶥ|ﶦ|ﶧ|ﶨ|ﶩ|ﶪ|ﶫ|ﶬ|ﶭ|ﶮ|ﶯ|ﶰ|ﶱ|ﶲ|ﶳ|ﶴ|ﶵ|ﶶ|ﶷ|ﶸ|ﶹ|ﶺ|ﶻ|ﶼ|ﶽ|ﶾ|ﶿ|ﷀ|ﷁ|ﷂ|ﷃ|ﷄ|ﷅ|ﷆ|ﷇ|ﷰ|ﷱ|ﷲ|ﷳ|ﷴ|ﷵ|ﷶ|ﷷ|ﷸ|ﷹ|ﷺ|ﷻ|﷼|﷽|ﹰ|ﹱ|ﹲ|ﹳ|ﹴ|﹵|ﹶ|ﹷ|ﹸ|ﹹ|ﹺ|ﹻ|ﹼ|ﹽ|ﹾ|ﹿ|ﻀ|ﻁ|ﻂ|ﻃ|ﻄ|ﻅ|ﻆ|ﻇ|ﻈ|ﻉ|ﻊ|ﻋ|ﻌ|ﻍ|ﻎ|ﻏ|ﻐ|ﻑ|ﻒ|ﻓ|ﻔ|ﻕ|ﻖ|ﻗ|ﻘ|ﻙ|ﻚ|ﻛ|ﻜ|ﻝ|ﻞ|ﻟ|ﻠ|ﻡ|ﻢ|ﻣ|ﻤ|ﻥ|ﻦ|ﻧ|ﻨ|ﻩ|ﻪ|ﻫ|ﻬ|ﻭ|ﻮ|ﻯ|ﻰ|ﻱ|ﻲ|ﻳ|ﻴ|ﻵ|ﻶ|ﻷ|ﻸ|ﻹ|ﻺ|ﻻ|ﻼ|﻽|﻾||！|＂|＃|＄|％|＆|＇|（|）|＊|＋|，|－|．|／|０|１|２|３|４|５|６|７|８|９|：|；|＜|＝|＞|？|＠|Ａ|Ｂ|Ｃ|Ｄ|Ｅ|Ｆ|Ｇ|Ｈ|Ｉ|Ｊ|Ｋ|Ｌ|Ｍ|Ｎ|Ｏ|Ｐ|Ｑ|Ｒ|Ｓ|Ｔ|Ｕ|Ｖ|Ｗ|Ｘ|Ｙ|Ｚ|［|＼|］|＾|＿|｀|ａ|ｂ|ｃ|ｄ|ｅ|ｆ|ｇ|ｈ|ｉ|ｊ|ｋ|ｌ|ｍ|ｎ|ｏ|ｐ|ｑ|ｒ|ｓ|ｔ|ｕ|ｖ|ｗ|ｘ|ｙ|ｚ|｛|｜|｝|～|｟|｠|｡|｢|｣|､|･|ｦ|ｧ|ｨ|ｩ|ｪ|ｫ|ｬ|ｭ|ｮ|ｯ|ｰ|ｱ|ｲ|ｳ|ｴ|ｵ|ｶ|ｷ|ｸ|ｹ|ｺ|ｻ|ｼ|ｽ|ｾ|ｿ|ﾀ|ﾁ|ﾂ|ﾃ|ﾄ|ﾅ|ﾆ|ﾇ|ﾈ|ﾉ|ﾊ|ﾋ|ﾌ|ﾍ|ﾎ|ﾏ|ﾐ|ﾑ|ﾒ|ﾓ|ﾔ|ﾕ|ﾖ|ﾗ|ﾘ|ﾙ|ﾚ|ﾛ|ﾜ|ﾝ|ﾞ|ﾟ|ﾠ|ﾡ|ﾢ|ﾣ|ﾤ|ﾥ|ﾦ|ﾧ|ﾨ|ﾩ|ﾪ|ﾫ|ﾬ|ﾭ|ﾮ|ﾯ|ﾰ|ﾱ|ﾲ|ﾳ|ﾴ|ﾵ|ﾶ|ﾷ|ﾸ|ﾹ|ﾺ|ﾻ|ﾼ|ﾽ|ﾾ|﾿|ￂ|ￃ|ￄ|ￅ|ￆ|ￇ|ￊ|ￋ|ￌ|ￍ|ￎ|ￏ|ￒ|ￓ|ￔ|ￕ|ￖ|ￗ|ￚ|ￛ|ￜ];
! Aspirated Consonants
define ASPICONSONANTS [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن|و] ھ];
! Aspirated Consonants with Shad
define ASPISHADCONSONANTS [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن|و] ّ ھ];
! Aspirated Consonants with ZABAR
define ASPIZABARCONSONANTS [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن|و] َ ھ];
! Aspirated Consonants with ZER
define ASPIZERCONSONANTS [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن|و] ِ ھ];
! Aspiracted Consonants with PESH
define ASPIPESHCONSONANTS [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن|و] ُ ھ];
! Aspirated Consonants with SHAD + ZABAR
! [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن] ّ َ ھ] | [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن] ّ َ ھ]
define ASPISHADZABARCONSONANTS [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن|و] ّ َ ھ];
! Aspirated Consonants with SHAD + ZER
![[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن] ّ ِ ھ] | [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن] ّ ِ ھ]
define ASPISHADZERCONSONANTS [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن|و] ّ ِ ھ];
! Aspiracted Consonants with SHAD + PESH
define ASPISHADPESHCONSONANTS [[ب|پ|ت|ٹ|ج|چ|د|ڈ|ر|ڑ|ک|گ|ل|م|ن|و] ّ ُ ھ];
! English Numbers
define ENGNUMBER [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9];
! Urdu Numbers
define URDNUMBER [۰ | ۱ | ۲ | ۳ | ۴ | ۵ | ۶ | ۷ | ۸ | ۹];
define ENGNUMBEREXPRESSION [ [ENGNUMBER [ENGNUMBER]*] ];
define URNUMBEREXPRESSION [ [URDNUMBER [URDNUMBER]*] ];
!*******************
! Diacritics
!*******************
! Shahmukhi Diacritics ٰ ٖ ٗ ٔ ٕ ٘ ٙ ٚ ٛ ٜ ٝ ٞ ٟ
! here we will ignore Mada Sign, Hamza Above, Hamza Below, and Noong-huna Mark diacritics
read regex [ َ -> 0, ِ -> [I 1], ُ -> [U 1], ٰ -> [A 3], ٖ -> [i 2], ٗ -> [u 2], ّ -> ["@" n 1], ٔ -> [I 1 n 1], ٕ,"."<- ٔ -> [U 1 n 1], ٘,0<- ٖ,0<- ٗ 0<- ُ,0<-];
!*******************
! Consonants
!*******************
! Simple Consonants
! 1st Hamza is Hamze+Chotiyeh Ligature and 2nd Hamza is hamza itslef
read regex [ ب -> b, پ -> p, ت -> [t "_" d], ٹ -> [t "`"], ث -> [s 1], ج -> [d "_" Z], چ -> [t "_" S], ح -> [h 1], خ -> [x], د -> [d "_" d], ڈ -> [d "`"], ذ -> [z 1], ر -> r, ڑ -> [r "`"], ز -> z, ژ -> Z, س -> s, ش -> S, ص -> [s 2], ض -> [z 2], ط -> [t "_" d 1], ظ -> [z 3],
```

```
ف -> f, ق -> q, ک -> k, گ -> g, ل -> l, م -> m, و -> v, ہ -> h, ء ->
[I 2], ئ -> [I 2], ی -> j, غ -> G, ۃ -> [t "_" d 2], ں -> [n "`"], ھ ->
h, ب -> [b 1], چ -> [d "_" Z 1], ڈ -> [d "`" 1], گ -> [g 1]];
! Simple Consonants + SHAD
read regex [ [ ّب] -> [b "." b], [ ّپ] -> [p "." p], [ ّت] -> [t "_" d
"." t "_" d], [ ّٹ] -> [t "`" "." t "`"], [ ّث] -> [s 1 "." s 1], [ ّج] -
> [d "_" Z "." d "_" Z], [ ّچ] -> [t "_" S "." t "_" S], [ ّح] -> [h 1
"." h 1], [ ّخ] -> [x "." x], [ ّد] -> [d "_" d "." d "_" d], [ ّڈ] -> [d
"`" "." d "`"], [ ّذ] -> [z 1 "." z 1], [ ّر] -> [r "." r], [ ّڑ] -> [r
"`" "." r "`"], [ ّز] -> [z "." z], [ ّژ] -> [Z "." Z], [ ّس] -> [s "."
s], [ ّش] -> [S "." S], [ ّص] -> [s 2 "." s 2], [ ّض] -> [z 2 "." z 2],
[ ّط] -> [t "_" d 1 "." t "_" d 1], [ ّظ] -> [z 3 "." z 3], [ ّف] -> [f
"." f], [ ّق] -> [q "." q], [ ّک] -> [k "." k], [ ّگ] -> [g "." g], [ ّل]
-> [l "." l], [ ّم] -> [m "." m], [ ّہ] -> [h "." h], [ ّغ] -> [G "." G],
[ ّں] -> [n "`" "." n "`"], [ ّھ] -> [h "." h], [ ّۃ] -> [t "_" d 2 "." t
"_" d 2], [ ّپ] -> [b 1 "." b 1], [ ّچ] -> [d "_" Z 1 "." d "_" Z 1], [ ّڈ
] -> [d "`" 1 "." d "`" 1], [ ّگ] -> [g 1 "." g 1]];
! Aspirated Consonants
read regex [ ھ -> ["_" h] || [م|ل|گ|ک|ڑ|ر|ڈ|د|چ|ج|ٹ|ت|پ|ب | v] _
];
! Aspirated Consonants + ZABAR
read regex [ [ب َھ] -> [b "_" h], [پ َھ] -> [p "_" h], [ت َھ] -> [t "_"
d "_" h], [ٹ َھ] -> [t "`" "_" h], [ج َھ] -> [d "_" Z "_" h], [چ َھ] ->
[t "_" S "_" h], [د َھ] -> [d "_" d "_" h], [ڈ َھ] -> [d "`" "_" h], [ر َ
ھ] -> [r "_" h], [ڑ َھ] -> [r "`" "_" h], [ک َھ] -> [k "_" h], [گ َھ] ->
[g "_" h], [ل َھ] -> [l "_" h], [م َھ] -> [m "_" h]];
! Aspirated Consonants + ZER
read regex [ [ب ِھ] -> [b "_" h I 1], [پ ِھ] -> [p "_" h I 1], [ت ِھ] ->
[t "_" d "_" h I 1], [ٹ ِھ] -> [t "`" "_" h I 1], [ج ِھ] -> [d "_" Z "_"
h I 1], [چ ِھ] -> [t "_" S "_" h I 1], [د ِھ] -> [d "_" d "_" h I 1], [ڈ
ِھ] -> [d "`" "_" h I 1], [ر ِھ] -> [r "_" h I 1], [ڑ ِھ] -> [r "`" "_"
h I 1], [ک ِھ] -> [k "_" h I 1], [گ ِھ] -> [g "_" h I 1], [ل ِھ] -> [l
"_" h I 1], [م ِھ] -> [m "_" h I 1]];
! Aspirated Consonants + ZER to ignore ZER while transliteration when
followed by CHOTI_YEH
read regex [ [ب ِھ] -> [b "_" h], [پ ِھ] -> [p "_" h], [ت ِھ] -> [t "_"
d "_" h], [ٹ ِھ] -> [t "`" "_" h], [ج ِھ] -> [d "_" Z "_" h], [چ ِھ] ->
[t "_" S "_" h], [د ِھ] -> [d "_" d "_" h], [ڈ ِھ] -> [d "`" "_" h], [ر ِ
ھ] -> [r "_" h], [ڑ ِھ] -> [r "`" "_" h], [ک ِھ] -> [k "_" h], [گ ِھ] -
> [g "_" h], [ل ِھ] -> [l "_" h], [م ِھ] -> [m "_" h] || _ i];
! Aspirated Consonants + PESH
read regex [ [ب ُھ] -> [b "_" h U 1], [پ ُھ] -> [p "_" h U 1], [ت ُھ] ->
[t "_" d "_" h U 1], [ٹ ُھ] -> [t "`" "_" h U 1], [ج ُھ] -> [d "_" Z "_"
h U 1], [چ ُھ] -> [t "_" S "_" h U 1], [د ُھ] -> [d "_" d "_" h U 1], [ڈ
ُھ] -> [d "`" "_" h U 1], [ر ُھ] -> [r "_" h U 1], [ڑ ُھ] -> [r "`" "_"
h U 1], [ک ُھ] -> [k "_" h U 1], [گ ُھ] -> [g "_" h U 1], [ل ُھ] -> [l
"_" h U 1], [م ُھ] -> [m "_" h U 1]];
! Aspirated Consonants + PESH to ignore PESH while transliteration
when followed by WAW
read regex [ [ب ُھ] -> [b "_" h], [پ ُھ] -> [p "_" h], [ت ُھ] -> [t "_"
d "_" h], [ٹ ُھ] -> [t "`" "_" h], [ج ُھ] -> [d "_" Z "_" h], [چ ُھ] ->
[t "_" S "_" h], [د ُھ] -> [d "_" d "_" h], [ڈ ُھ] -> [d "`" "_" h], [ر ُ
```

```
ھ] -> [r "_" h], [ھُ ڑ] -> [r "`" "_" h], [ھُ ک] -> [k "_" h], [ھُ گ] -
> [g "_" h], [ھُ ل] -> [l "_" h], [ھُ م] -> [m "_" h] || _ u];
! Aspirated Consonants + SHAD
read regex [ [ھ ّ ب] -> [b "." b "_" h], [ھ ّ پ] -> [p "." p "_" h], [ ّ ت
ھ] -> [t "_" d "." t "_" d "_" h], [ھ ّ ٹ] -> [t "`" "." t "`" "_" h],
[ھ ّ ج] -> [d "_" Z "." d "_" Z "_" h], [ھ ّ چ] -> [t "_" S "." t "_" S
"_" h], [ھ ّ د] -> [d "_" d "." d "_" d "_" h], [ھ ّ ڈ] -> [d "`" "." d "`"
"_" h], [ھ ّ ر] -> [r "." r "_" h], [ھ ّ ڑ] -> [r "`" "." r "`" "_" h], [ک
ھ ّ] -> [k "." k "_" h], [ھ ّ گ] -> [g "." g "_" h], [ھ ّ ل] -> [l "." l
"_" h], [ھ ّ م] -> [m "." m "_" h], [ ّ ھ ب] -> [b "." b "_" h], [ ّ ھ پ] ->
[p "." p "_" h], [ ّ ھ ت] -> [t "_" d "." t "_" d "_" h], [ ّ ھ ٹ] -> [t
"`" "." t "`" "_" h], [ ّ ھ ج] -> [d "_" Z "." d "_" Z "_" h], [ ّ ھ چ] ->
[t "_" S "." t "_" S "_" h], [ ّ ھ د] -> [d "_" d "." d "_" d "_" h], [ ڈ ھ
ّ] -> [d "`" "." d "`" "_" h], [ ّ ھ ر] -> [r "." r "_" h], [ ّ ھ ڑ] -> [r
"`" "." r "`" "_" h], [ ّ ھ ک] -> [k "." k "_" h], [ ّ ھ گ] -> [g "." g
"_" h], [ ّ ھ ل] -> [l "." l "_" h], [ ّ ھ م] -> [m "." m "_" h]];
! Aspirated Consonants + SHAD + ZABAR
read regex [ [ھ َ ّ ب] -> [b "." b "_" h], [ھ َ ّ پ] -> [p "." p "_" h], [ ت
ھ َ ّ] -> [t "_" d "." t "_" d "_" h], [ھ َ ّ ٹ] -> [t "`" "." t "`" "_"
h], [ھ َ ّ ج] -> [d "_" Z "." d "_" Z "_" h], [ھ َ ّ چ] -> [t "_" S "." t
"_" S "_" h], [ھ َ ّ د] -> [d "_" d "." d "_" d "_" h], [ھ َ ّ ڈ] -> [d "`"
"." d "`" "_" h], [ھ َ ّ ر] -> [r "." r "_" h], [ھ َ ّ ڑ] -> [r "`" "." r
"`" "_" h], [ھ َ ّ ک] -> [k "." k "_" h], [ھ َ ّ گ] -> [g "." g "_" h], [ ّ ل
ھ َ] -> [l "." l "_" h], [ھ َ ّ م] -> [m "." m "_" h]];
! Aspirated Consonants + SHAD + ZER
read regex [ [ھ ِ ّ ب] -> [b "." b "_" h I 1], [ھ ِ ّ پ] -> [p "." p "_" h I
1], [ھ ِ ّ ت] -> [t "_" d "." t "_" d "_" h I 1], [ھ ِ ّ ٹ] -> [t "`" "." t
"`" "_" h I 1], [ھ ِ ّ ج] -> [d "_" Z "." d "_" Z "_" h I 1], [ھ ِ ّ چ] ->
[t "_" S "." t "_" S "_" h I 1], [ھ ِ ّ د] -> [d "_" d "." d "_" d "_" h I
1], [ھ ِ ّ ڈ] -> [d "`" "." d "`" "_" h I 1], [ھ ِ ّ ر] -> [r "." r "_" h I
1], [ھ ِ ّ ڑ] -> [r "`" "." r "`" "_" h I 1], [ھ ِ ّ ک] -> [k "." k "_" h I
1], [ھ ِ ّ گ] -> [g "." g "_" h I 1], [ھ ِ ّ ل] -> [l "." l "_" h I 1], [ ِ ّ م
ھ] -> [m "." m "_" h I 1]];
! Aspirated Consonants + SHAD + ZER while transliteration when fol-
lowed by CHOTI_YEH
read regex [ [ھ ِ ّ ب] -> [b "." b "_" h], [ھ ِ ّ پ] -> [p "." p "_" h], [ ت
ھ ِ ّ] -> [t "_" d "." t "_" d "_" h], [ھ ِ ّ ٹ] -> [t "`" "." t "`" "_"
h], [ھ ِ ّ ج] -> [d "_" Z "." d "_" Z "_" h], [ھ ِ ّ چ] -> [t "_" S "." t
"_" S "_" h], [ھ ِ ّ د] -> [d "_" d "." d "_" d "_" h], [ھ ِ ّ ڈ] -> [d "`"
"." d "`" "_" h], [ھ ِ ّ ر] -> [r "." r "_" h], [ھ ِ ّ ڑ] -> [r "`" "." r
"`" "_" h], [ھ ِ ّ ک] -> [k "." k "_" h], [ھ ِ ّ گ] -> [g "." g "_" h], [ ّ ل
ھ ِ] -> [l "." l "_" h], [ھ ِ ّ م] -> [m "." m "_" h] || _ i];
! Aspirated Consonants + SHAD + PESH
read regex [ [ھ ُ ّ ب] -> [b "." b "_" h U 1], [ھ ُ ّ پ] -> [p "." p "_" h U
1], [ھ ُ ّ ت] -> [t "_" d "." t "_" d "_" h U 1], [ھ ُ ّ ٹ] -> [t "`" "." t
"`" "_" h U 1], [ھ ُ ّ ج] -> [d "_" Z "." d "_" Z "_" h U 1], [ھ ُ ّ چ] ->
[t "_" S "." t "_" S "_" h U 1], [ھ ُ ّ د] -> [d "_" d "." d "_" d "_" h U
1], [ھ ُ ّ ڈ] -> [d "`" "." d "`" "_" h U 1], [ھ ُ ّ ر] -> [r "." r "_" h U
1], [ھ ُ ّ ڑ] -> [r "`" "." r "`" "_" h U 1], [ھ ُ ّ ک] -> [k "." k "_" h U
1], [ھ ُ ّ گ] -> [g "." g "_" h U 1], [ھ ُ ّ ل] -> [l "." l "_" h U 1], [ ُ ّ م
ھ] -> [m "." m "_" h U 1]];
! Aspirated Consonants + SHAD + PESH while transliteration when fol-
lowed by WAW
```

```
read regex [ [ھ ُ ّ ب] -> [b "." b "_" h], [ھ ُ ّ پ] -> [p "." p "_" h], [ت
ھ ُ ّ] -> [t "_" d "." t "_" d "_" h], [ھ ُ ّ ٹ] -> [t "`" "." t "`" "_"
h], [ھ ُ ّ ج] -> [d "_" Z "." d "_" Z "_" h], [ھ ُ ّ چ] -> [t "_" S "." t
"_" S "_" h], [ھ ُ ّ د] -> [d "_" d "." d "_" d "_" h], [ھ ُ ّ ڈ] -> [d "`"
"." d "`" "_" h], [ھ ُ ّ ر] -> [r "." r "_" h], [ھ ُ ّ ڑ] -> [r "`" "." r
"`" "_" h], [ھ ُ ّ ک] -> [k "." k "_" h], [ھ ُ ّ گ] -> [g "." g "_" h], [ّ ل
ھ ُ] -> [l "." l "_" h], [ھ ُ ّ م] -> [m "." m "_" h] || _ u];
!******************
! Gol Heh at the end of a word
! Gol Heh is considered as a vowel A1 when it is at the end of a word
and is preceeded by a consonant. we will not convert it into A1, as
when we need to convert the text back in Shahmukhi, it will produce
ambiguity. we will handle this issue when we will convert UIT into
Gurmukhi. Two Gol Hehs at the end of word are considered as consonant
'h'. Or a Gol heh is considered as consonant 'h' when it is at the end
of word and is preceeded by a vowel.
! ******************
! Short Vowels with consonants
! WAW as vowel o after a consonant
read regex [و -> [o 1] || [CONSONANTS | ASPICONSONANTS | ASPISHADCONSO-
NANTS | j | v | n] _ ];
! ZABAR + WAW as au after a consonant
read regex [[ً و] -> [O 1] || [CONSONANTS | ASPICONSONANTS | ASPISHAD-
CONSONANTS | j | v | n] _ ];
! PESH + WAW as u after a consonant
read regex [[ُ و] -> [u 1] || [CONSONANTS | ASPICONSONANTS | ASPISHAD-
CONSONANTS | j | v | n] _ ];
! WAW as au after a consonant
read regex [و -> [O 1] || [ASPIZABARCONSONANTS | ASPISHADZABARCONSO-
NANTS] _ ];
! WAW as u after a consonant
read regex [و -> [u 1] || [ASPIPESHCONSONANTS | ASPISHADPESHCONSONANTS]
_ ];
! YEH as vowel e after a consonant
read regex [ ی -> [e 1] || [CONSONANTS | ASPICONSONANTS | ASPISHADCON-
SONANTS | j | v | n] _ ];
! ZABAR + YEH as vowel ai after a consonant
read regex [[ً 1 "}"] <- [ی] || [CONSONANTS | ASPICONSONANTS | ASPISHAD-
CONSONANTS | j | v | n] _ ];
! ZER + YEH as vowel i after a consonant
read regex [[ِ ی] -> [i 1] || [CONSONANTS | ASPICONSONANTS | ASPISHAD-
CONSONANTS | j | v | n] _ ];
! YEH as i after a consonant
read regex [ی -> [i 1] || [ASPIZERCONSONANTS | ASPISHADZERCONSONANTS]
_ ];
! YEH as i after a consonant
read regex [1 "}"] <- ی] || [ASPIZABARCONSONANTS | ASPISHADZABARCONSO-
NANTS] _ ];
!******************
! Vowels
!******************
! Long Vowels not at the start of a word
! ******************
! Cases of Bari YEH
! Bari Yeh not at the end of a word
read regex [ے -> [e 4]];
! Bari YEH after ZABAR and not at the end of a word
read regex [ً 4 "}"] <- ے]];
```

```
read regex [[ےء] -> [I 2 e 4], [ےئ] -> [I 2 e 4]];
read regex [4 "}"] <- ے] || [ASPIZABARCONSONANTS | ASPISHADZABARCONSO-
NANTS] _ ];
! Bari yeh will always form the vowel with a number 4 in the middle or
at the end of a word
! Vowels with AIN any where else
read regex [ ع -> ["?"] ];
read regex [ [َع] -> ["?"] ];
read regex [ [ِع] -> ["?" I 1], [ُع] -> ["?" U 1]];
read regex [ [ىع] -> ["?" e 1], [ىِع] -> ["?" i 1], [ ,[1 "}" "?"] <- [ىَع
وع]] -> ["?" o 1], [ وَع] -> ["?" O 1], [وُع] -> ["?" u 1], [ےع] ->
["?" e 4], [4 "}" "?"] <- [ےَع]];
! Vowels with ALEF any where else
read regex [ آ -> A];
read regex [ ا -> [A 1] ];
read regex [ [َا] -> [A 1] ];
read regex [ [ِا] -> I, [ُا] -> U];
read regex [ [ىا] -> e, [ىِا] -> i, [وا],"}" <- [ىَا] -> o, [ وَا] -> O, [ا
وُ] -> u, [ےا] -> [e 3], [3 "}"] <- [ےَا]];
! Vowels with AIN at the start of a word or a syllable
read regex [ ْ|.#.] || "?" <- ع | " "] _ ];
read regex [ [ْ|.#.] || "?" <- [َع | " "] _ ];
read regex [ [ِع] -> ["?" I 1], [ُع] -> ["?" U 1] || [.#. | ْ | " "] _
];
read regex [ [ىع] -> ["?" e 1], [ىِع] -> ["?" i 1], [ ,[1 "}" "?"] <- [ىَع
وع]] -> ["?" o 1], [ وَع] -> ["?" O 1], [وُع] -> ["?" u 1], [ےع] ->
["?" e 4], [ْ|.#.] || [4 "}" "?"] <- [ےَع | " "] _ ];
! Vowels with ALEF at the start of a word or a syllable
read regex [ آ -> A || [.#. | ْ | " "] _ ];
read regex [ ْ|.#.] || ["@"] <- ا | " "] _ ];
read regex [ [ْ|.#.] || ["@"] <- [َا | " "] _ ];
read regex [ [ِا] -> I, [ُا] -> U || [.#. | ْ | " "] _ ];
read regex [ [ىا] -> e, [ىِا] -> i, [وا],"}" <- [ىَا] -> o, [ وَا] -> O, [ا
وُ] -> u, [ےا] -> [e 3], [ْ|.#.] || [3 "}"] <- [ےَا | " "] _ ];
! ********************
! NOONGHUNA
read regex [ں -> "~"];
! ********************
! NOON when it is not considered as NOONGHUNA
! NOON as a nasalization
read regex [ن -> "~"];
! Noon followed by Noonghuna mark
read regex [[ّ ن] -> "~"];
! Noon + diacrictical marks
read regex [ن -> n || _ [ َ|ٰ|ً|ْ|ٍ|ٌ|ُ|ِ]];
read regex [[ّن] -> [n "." n]];
! NOON followed by Heh Doachashmi as an aspirated consonant
read regex [[ھن] -> [n "_" h]];
! Noon + Diacritics+ Heh Doachashmi
read regex [[ھَن] -> [n "_" h َ]];
read regex [[ھِن] -> [n "_" h I 1]];
read regex [[ھِن] -> [n "_" h] || _ [i 1 .#.]];
read regex [[ھِن] -> [n "_" h ی _ || [ِ]];
read regex [[ھُن] -> [n "_" h U 1]];
```

```
read regex [[ھ ُ ن] -> [n "_" h و _ || [ ُ];
read regex [[ھ ّ ن] -> [n "." n "_" h]];
read regex [[ھ َ ّ ن] -> [n "." n "_" h َ]];
read regex [[ھ ِ ّ ن] -> [n "." n "_" h I 1]];
read regex [[ھ ِ ّ ن] -> [n "." n "_" h] || _ [i 1 .#.]];
read regex [[ھ ِ ّ ن] -> [n "." n "_" h ی _ || [ ِ];
read regex [[ھ ُ ّ ن] -> [n "." n "_" h U 1]];
read regex [[ھ ُ ّ ن] -> [n "." n "_" h و _ || [ ُ];
! Noon + Heh Doachashmi + Diacritics
read regex [[ ِ ھ ن] -> [n "_" h] || _ [i 1 .#.]];
read regex [[ ّ ھ ن] -> [n "." n "_" h]];
read regex [[ ِ ّ ھ ن] -> [n "." n "_" h] || _ [i 1 .#.]];
! NOON before ALEF, ALEF Mada, Bari YEH, and diacritics
read regex [ن -> n || _ [ ا|آ|ے|ع|ن | i | ء|ئ|و|ی | v | j | I]];
! NOON at the end of a word
read regex [ن -> n || _ [ہ [.#. | " "]] ];
read regex [ن -> n || _ [.#. | ْ | " "] ];
! NOON at the start of a word
read regex [ن -> n || [.#. | ْ | " "] _ ];
!*******************
! Special Cases for Ain with Alef
!*******************
read regex [ع -> "?" || _ [i]];
read regex [[ع ا] -> [A 1 "?"] || ? _ ];
read regex [[ْ|.#.] || ["?" "@"] <- [ع ا | " "] _ ];
read regex [[ع ا] -> ["?" A 1]];
!*******************
! Special Cases for Waw
!*******************
read regex [و -> [o 1], ا -> [A 1], آ -> A, ع -> "?" || [I | 1 | 4] _ ];
!*******************
! Special Cases for Yeh
!*******************
! Yeh before Alef or Waw as I vowel
read regex [ی -> I || _ [ا|آ|و|ع]];
read regex [ی -> [I 1] || [CONSONANTS | ASPICONSONANTS | ن | v | j] _
[ا|آ|و|ع]];
! Yeh before Gol heh at the end of a word
read regex [ی -> I || _ [ہ [.#. | " "]]];
read regex [ی -> [I 1] || [CONSONANTS | ASPICONSONANTS | v | j] _ [ہ [
.#. | " " ]]];
رباعیاں داعیوں !
!**********read regex [ی -> [i 1] || [ا|آ|و] _ [ع [ا|ی]]];
!*******************
! When Waw will be considered as a consonant
! Waw is followed by a vowel
read regex [و -> v || _ [ا|آ|ی|ے|ً|ٌ|ٍ|َ|ُ|ِ|ّ|ْ|ٔ | i | ھ]];
read regex [و -> v || [ا|ع|آ|ی|و] _ [ا|ع|آ|ی|و]];
read regex [[ّ و] -> [v "." v]];
! Waw + Diacritics+ Heh Doachashmi
read regex [[ھ َ و] -> [v "_" h َ]];
read regex [[ھ ِ و] -> [v "_" h I 1]];
read regex [[ھ ِ و] -> [v "_" h] || _ [i 1 .#.]];
read regex [[ھ ِ و] -> [v "_" h ی _ || [ ِ];
read regex [[ھ ُ و] -> [v "_" h U 1]];
```

```
read regex [[ھ ُ و] -> [v "_" h و _ || [ُ]];
read regex [[ھ ّ و] -> [v "." v "_" h]];
read regex [[ھ ّ َ و] -> [v "." v "_" h َ]];
read regex [[ھ ِ ّ و] -> [v "." v "_" h I 1]];
read regex [[ھ ِ ّ و] -> [v "." v "_" h] || _ [i 1 .#.]];
read regex [[ھ ِ ّ و] -> [v "." v "_" h ی _ || [ِ]];
read regex [[ھ ُ ّ و] -> [v "." v "_" h U 1]];
read regex [[ھ ُ ّ و] -> [v "." v "_" h و _ || [ُ]];
! Noon + Heh Doachashmi + Diacritics
read regex [[ھ ِ و] -> [v "_" h] || _ [i 1 .#.]];
read regex [[ھ ّ و] -> [v "." v "_" h]];
read regex [[ھ ِ ّ و] -> [v "." v "_" h] || _ [i 1 .#.]];
! Waw is followed by Shad
! چوّا، بوّا، سوّا
read regex [[ّ و] -> [U 1 v] || [CONSONANTS | ASPICONSONANTS | ASPISHAD-
CONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPISHADZABAR-
CONSONANTS | ASPISHADPESHCONSONANTS | j] _ [ا]];
! Waw at the start of a word
read regex [و -> v || [.#. | ٔ | " "] _ ];
!*******************
! When Yeh will be considered as a consonant
read regex [[ّ ی] -> [j "." j]];
read regex [[ّ ی] -> [i 1 j] || [CONSONANTS | ASPICONSONANTS | ASPI-
SHADCONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPISHAD-
ZABARCONSONANTS | ASPISHADPESHCONSONANTS] _ ];
read regex [[ً ی] -> [i 1 j] || [ASPIZERCONSONANTS | ASPISHADZERCONSO-
NANTS] _ ];
read regex [[ِ ّ ی] -> [i 1 j] || [CONSONANTS | ASPICONSONANTS | ASPI-
SHADCONSONANTS | ASPIZABARCONSONANTS | ASPIPESHCONSONANTS | ASPISHAD-
ZABARCONSONANTS | ASPISHADPESHCONSONANTS] _ ];
! Yeh in between of two Alefs
! ایام، عیال، عیاں
read regex [ی -> j || [[.#. | و|ا] _ [[ع|ا] ["ٔ"|ٰ]];
! Yeh followed by Shad after Alef or Ain
read regex [[ّی] -> [j "." j] || [ع|ا] _ ];
! اعیان
read regex [ی -> j || [[.#. | ا] _ [[ع ا] ["ٔ"|ٰ]];
! Yeh is followed by a diacritic
read regex [ی -> j || _ [ ْ|ٖ|ِ|ُ|َ|ٍ|ٌ|ً]];
! Yeh at the start of a word
read regex [ی -> j || [.#. | ٔ | " "] _ ];
! Yeh at the end of a word
read regex [ی -> [i 1] || _ [.#. | ٔ | " "]];
!*******************
! Special Cases
!*******************
! Special Cases for Hamza Waw
!*******************
read regex [[ئو] -> [I 2 o 1] , [ءو] -> [I 2 o 1]];
! Hamza + Waw gives sound of long vowel 'u' when it comes 2nd last in
the word and the word is a verb. جائوں، جھلوائوں، جیئوں
! In case, the word is a noun, the Hamza + waw gives the sound of 'o'.
so solve this problem, we need POS tagging of words.
read regex [[ئو] -> [I 2 u 1] , [ءو] -> [I 2 u 1] || _ [ن|ں]];
! ******************
! HAMZA after YEH as "@" vowel sound
```

```
read regex [[ءی] -> [I 2 i 1], [ئی] -> [I 2 i 1]];
! It seems that Hamza + Waw is never followed by Choti-Yeh. only word
that exist is سائویں (I think that it is wrong spelling of ساویں)
! Special strings at the end of a word
read regex [ [ٰی] -> A 2, [ٰی] -> A 2];
! First Hamza is Hamza and second is Hamza + Yeh Ligature
read regex [ [ٰی] -> A 2, [ٰی] -> A 2, [ًا] -> ["@" n], [ءی] -> [I 2 i
1], [ئی] -> [I 2 i 1], [ءو] -> [I 2 o 1], [ئو] -> [I 2 o 1]  || _
[.#. |ْ| " "] ];
! ******************
! Compound Words
read regex [ ِ -> ["-" e "-"], ء -> ["-" e 1 "-"] || [CONSONANTS | AS-
PICONSONANTS] _ " "];
!******************
! Vowels alone
read regex [ْ|.#.] _ [" "|ْ|.#.] || "@" <-ا| " "]];
read regex [[ِا],"@" <-[َا] -> I, [ُا] -> U || [.#. |ْ|.#.] _ [" "|ْ| " "]];
read regex [[ای] -> i, [ِای] -> i, [او],["}"] <-[َای] -> o, [َاو] -> O, [ُا
و] -> u, [اے] -> [e 3], [ْ|.#.] _ [" "|ْ|.#.] || [3 "}"] <-[َاے| " "] ];
!******************
! Special words
read regex [[لله] -> ["@" l "." l A 1 h] || [? - ا] _ ];
read regex [[الله] -> ["@" l "." l A 1 h], [الله] -> ["@" l "." l A 1
h], [اللّٰه] -> ["@" l "." l A 1 h], [اللّه] -> ["@" l "." l A 1 h]];
read regex [[کہ] -> [k I 1], [وہ] -> [v h], [یہ] -> [j h] || [.#. | "
"] _ [.#. | " "]];
!******************
! Number
!******************
read regex [۰ -> "+URZERO", ۱ -> "+URONE", ۲ -> "+URTWO", ۳ ->
"+URTHREE", ۴ -> "+URFOUR", ۵ -> "+URFIVE", ۶ -> "+URSIX", ۷ ->
"+URSEVEN", ۸ -> "+UREIGHT", ۹ -> "+URNINE"];
! Decimal and thousand separator
read regex [، -> ["+URDECIMAL"], ، -> "+URTHSEP"];
read regex ["+DOT" -> "+URDECIMAL", "+COMMA" -> "+URTHSEP" || URNUMBE-
REXPRESSION _ URNUMBEREXPRESSION];
read regex ["0" -> "+ZERO", 1 -> "+ONE", 2 -> "+TWO", 3 -> "+THREE", 4
-> "+FOUR", 5 -> "+FIVE", 6 -> "+SIX", 7 -> "+SEVEN", 8 -> "+EIGHT", 9
-> "+NINE"];
read regex ["+DOT" -> "+DECIMAL", "+COMMA" -> "+THSEP" || ENGNUMBEREX-
PRESSION _ ENGNUMBEREXPRESSION];
read regex ["." -> "+DOT", ؟ -> "+URQMARK", ۔ -> "+URSTBD", ،->
"+URCOMMA", "," -> "+COMMA", "+NL" -> "+NL", ٪-> "+URPERCENT", ؛ ->
"+URSEMICOLON", ٭ -> "+URSTAR", "" -> 0];
compose net
```

UIT to Seraiki/Shahmukhi Finite-state Transducer

```
clear stack
set char-encoding UTF-8
!******************
!Definition of Varibales
!******************
! CONSONANTS | ASPICONSONANTS
define CONSONANTS [b | [b "_" h] | p | [p "_" h] | [t "_" d] | [t "_"
d "_" h] | [t "`"] | [t "`" "_" h] | [s 1] | [d "_" Z] | [d "_" Z "_"
```

```
h] | [t "_" S] | [t "_" S "_" h] | [h 1] | x | [d "_" d] | [d "_" d
"_" h] | [d "`"] | [d "`" "_" h] | [z 1] | r | [r "_" h] | [r "`"] |
[r "`" "_" h] | z | Z| s | S | [s 2] | [z 2] | [t "_" d 1] | [z 3] | G
| f | q | k | [k "_" h] | g | [g "_" h] | l | [l "_" h] | m | [m "_"
h] | n | [n "_" h] | v | [v "_" h] | h | j | [t "_" d 2] | H | [N 1] |
[N 2] | [n "`"] | [l "`"] | [b 1] | [d "_" Z 1] | [d "`" 1] | [g 1]];
define VOWELS ["@" | A | I | U | e | "{" | i | u | O | o | [A 1] | [A
2] | [A 3] | [I 1] | [I 2] | [U 1] | [e 1] | ["{" 1] | [i 1] | [u 1] |
[O 1] | [o 1] | [e 3] | [e 4] | ["{" 3] | ["{" 4] | [i 2] | [u 2]];
!*******************
! Multi-character symbols
!*******************
read regex ["+DOT" -> ".", "+QMARK" -> "?", "+DECIMAL" -> ".",
"+COMMA" -> ",", "+THSEP" -> ",", "+SEMICOLON" -> ";", "+COLON" ->
":", "+PERCENT" -> ٪, "+ZERO" -> "0", "+ONE" -> 1, "+TWO" -> 2,
"+THREE" -> 3, "+FOUR" -> 4, "+FIVE" -> 5, "+SIX" -> 6, "+SEVEN" -> 7,
"+EIGHT" -> 8, "+NINE" -> 9, "+URZERO" -> ٠, "+URONE" -> ١, "+URTWO" -
> ٢, "+URTHREE" -> ٣, "+URFOUR" -> ۴, "+URFIVE" -> ۵, "+URSIX" -> ۶,
"+URSEVEN" -> ٧, "+UREIGHT" -> ٨, "+URNINE" -> ٩, "+URQMARK" -> ؟,
"+URSTBD" -> ٫, "+URCOMMA" -> ٬, "+URPERCENT" -> ٪, "+URSEMICOLON" -> ؛,
"+URSTAR" -> ٭, "+URDECIMAL" -> ٫, "+URTHSEP" -> ٬];
!*******************
! Diacritics
!*******************
read regex ["." -> ◌ٔ];
!*******************
! Consonants
!*******************
! Simple Consonants
! Consonants Set 1
read regex [b -> ب, p -> پ, [t "_" d] -> ت, [t "`"] -> ٹ, x -> خ, [d
"_" d] -> د, [d "`"] -> ڈ, r -> ر, z -> ز, Z -> ژ, s -> س, S -> ش, G -
> غ, f -> ف, q -> ق, k -> ک, g -> گ, l -> ل, m -> م, n -> ن, v -> و, h
-> ہ, j -> ی, H -> ھ, [N 1] -> [ن گ], [N 2] -> [ن ج]];
! Consonants Set 2
read regex [[s 1] -> ث, [d "_" Z] -> ج, [t "_" S] -> چ, [h 1] -> ح, [z
1] -> ذ, [r "`"] -> ڑ, [S 1] -> ش, [s 2] -> ص, [z 2] -> ض, [t "_" d
1] -> ط, [z 3] -> ظ, [t "_" d 2] -> ۃ, [n "`"] -> ں, [l "`"] -> ل, [b
1] -> پ, [d "`" 1] -> ڈ◌ٔ, [g 1] -> گ];
! Consonants Set 3
read regex [[d "_" Z 1] -> چ];
! Aspirated Consonants
read regex [ ["_" h] -> ھ];
! Germinated Simple Consonants
read regex [[b "." b] -> [◌ّب], [p "." p] -> [◌ّپ], [t "_" d "." t "_" d]
-> [◌ّت], [t "`" "." t "`"] -> [◌ّٹ], [s 1 "." s 1] -> [◌ّث], [d "_" Z "."
d "_" Z] -> [◌ّج], [t "_" S "." t "_" S] -> [◌ّچ], [h 1 "." h 1] -> [◌ّح],
[x "." x] -> [◌ّخ], [d "_" d "." d "_" d] -> [◌ّد], [d "`" "." d "`"] ->
[◌ّڈ], [z 1 "." z 1] -> [◌ّذ], [r "." r] -> [◌ّر], [r "`" "." r "`"] -> [◌ّ
ڑ], [z "." z] -> [◌ّز], [Z "." Z] -> [◌ّژ], [s "." s] -> [◌ّس], [S "." S] ->
[◌ّش], [S 1 "." S 1] -> [◌ّش], [s 2 "." s 2] -> [◌ّص], [z 2 "." z 2] ->
[◌ّض], [t "_" d 1 "." t "_" d 1] -> [◌ّط], [z 3 "." z 3] -> [◌ّظ], [G "."
G] -> [◌ّغ], [f "." f] -> [◌ّف], [q "." q] -> [◌ّق], [k "." k] -> [◌ّک], [g
"." g] -> [◌ّگ], [l "." l] -> [◌ّل], [m "." m] -> [◌ّم], [n "." n] -> [◌ّن],
[v "." v] -> [◌ّو], [h "." h] -> [◌ّہ], [j "." j] -> [◌ّی], [t "_" d 2 "." t
```

```
"_" d 2] -> [ّة], [n "`" "." n "`"] -> [ّڻ], [l "`" "." l "`"] -> [ّل],
[b 1 "." b 1] -> [ّپ], [d "_" Z 1 "." d "_" Z 1] -> [ّچ], [d "`" 1 "."
d "`" 1] -> [ّڈ], [g 1 "." g 1] -> [ّگ]];
! Germinated Aspirated Consonants
read regex [[b "." b "_" h] -> [ّب ھ], [p "." p "_" h] -> [ّپ ھ], [t
"_" d "." t "_" d "_" h] -> [ّت ھ], [t "`" "." t "`" "_" h] -> [ّٹ ھ],
[d "_" Z "." d "_" Z "_" h] -> [ّج ھ], [t "_" S "." t "_" S "_" h] ->
[ّچ ھ], [d "_" d "." d "_" d "_" h] -> [ّد ھ], [d "`" "." d "`" "_" h] -
> [ّڈ ھ], [r "." r "_" h] -> [ّر ھ], [r "`" "." r "`" "_" h] -> [ّڑ ھ],
[k "." k "_" h] -> [ّک ھ], [g "." g "_" h] -> [ّگ ھ], [l "." l "_" h] -
> [ّل ھ], [m "." m "_" h] -> [ّم ھ], [n "." n "_" h] -> [ّن ھ], [v "." v
"_" h] -> [ّو ھ]];
! ********************
! Nazalization
! ********************
read regex ["~" -> ں];
read regex [["~" 1] -> ں];
read regex ["~" -> ں || _ [.#. | " "]];
read regex [["~" 1] -> ں || _ [.#. | " "]];
!********************
! Vowels
!********************
! Vowels in the middle or at the end
read regex [A -> ا <- "@",آ, I -> [ِا], U -> [ُا], i -> [ِی ا], o -> [او], O
-> [اَو], u -> [اُو], e -> [اَی] <- [ "}"] ,[ای]];
read regex [[e 3] -> [ع <- "?",[اَے] <- [3 "}"],[اے]];
read regex [[A 1] -> ا, [I 1] -> ِ,[U 1] -> ُ, [i 1] -> [ِی], [o 1] ->
و, [O 1] -> [اَ], [u 1] -> [اُ], [e 1] -> ی َ] <- [1 "}"] ,ی], [e 4] -> ,ے
ے] <- [4 "}"]]];
read regex [["@" n] -> [ًا]];
read regex [[A 2] -> [ٰی], [A 3] -> ٰ, [I 2] -> [ٔ], [i 2] -> ٔ , [u 2]
-> ٗ, ["@" n 1] -> ً, [I 1 n 1] -> ٍ, [U 1 n 1] -> ٌ];
! ********************
! Special cases
! ********************
! Special cases of vowels
read regex [I -> ی || [VOWELS | "?"] _ [[h [ .#. | " " ]] | A | o |
"?"]];
read regex [[I 1] -> ی || CONSONANTS _ [[h [ .#. | " " ]] | A | o |
"?"]];
read regex [[U 1 v] -> [ّو] || CONSONANTS _ A];
read regex [[i 1 j] -> [ِّی] || CONSONANTS _ ];
read regex [[i 1] -> ی, [e 1] -> [ے] <- [1 "}"] ,[ے] || _ [.#. | " "]];
read regex [[I 2 i 1] -> [ئی], [I 2 e 1] -> [ئی], [I 2 "{" 1] -> [ئ
ی], [I 2 o 1] -> [ئو], [I 2 u 1] -> [ئو], [I 2 U 1] -> [ئو], [I 2 O 1]
-> [ئو], [I 2 e 4] -> [ئے]];
read regex [[I 2 i 1] -> [ئی] || ];
! ********************
! Compound Words
read regex [["-" e "-"] -> ِ, ["-" e 1 "-"] -> ہ || _ [" "]];
!********************
! Special words
```

```
read regex [[k I 1] -> [ہ ک] || [[.#. | " "] [[k j u 1 "~"] | [t "_" S
u 1 "~"] | [t "_" Z b] | [d "_" Z o 1] | [n A 1] | [b l]]] _ [.#. | "
"]];
read regex [ ["@" l "." l A 1 h] -> [ہ ںںال]];
read regex [[k I 1] -> [ہ ک] || [ .#. | " " ] _ [ .#. | " " ]];
! ******************
! Multi-character symbols
read regex ["+DOT" -> "DOT", "+QMARK" -> "+QMARK", "+DECIMAL" ->
"+DECIMAL", "+COMMA" -> "+COMMA", "+NL" -> "+NL", "+THSEP" ->
"+THSEP", "+SEMICOLON" -> "+SEMICOLON", "+COLON" -> "+COLON",
"+PERCENT" -> "+PERCENT", "+ZERO" -> "+ZERO", "+ONE" -> "+ONE", "+TWO"
-> "+TWO", "+THREE" -> "+THREE", "+FOUR" -> "+FOUR", "+FIVE" ->
"+FIVE", "+SIX" -> "+SIX", "+SEVEN" -> "+SEVEN", "+EIGHT" -> "+EIGHT",
"+NINE" -> "+NINE", "+DSTBD" -> "+URSTBD", "+SSTBD" -> "+URSTBD",
"+GURDECIMAL" -> "+URDECIMAL", "+GURTHSEP" -> "+URTHSEP", "+GURZERO" -
> "+URZERO", "+GURONE" -> "+URONE", "+GURTWO" -> "+URTWO", "+GURTHREE"
-> "+URTHREE", "+GURFOUR" -> "+URFOUR", "+GURFIVE" -> "+URFIVE",
"+GURSIX" -> "+URSIX", "+GURSEVEN" -> "+URSEVEN", "+GUREIGHT" ->
"+UREIGHT", "+GURNINE" -> "+URNINE", "+URQMARK" -> "+URQMARK",
"+URSTBD" -> "+URSTBD", "+URCOMMA" -> "+URCOMMA", "+URPERCENT" ->
"+URPERCENT", "+URSEMICOLON" -> "+URSEMICOLON", "+URSTAR" ->
"+URSTAR", "+URDECIMAL" -> "+URDECIMAL", "+URTHSEP" -> "+URTHSEP",
"+URZERO" -> "+URZERO", "+URONE" -> "+URONE", "+URTWO" -> "+URTWO",
"+URTHREE" -> "+URTHREE", "+URFOUR" -> "+URFOUR", "+URFIVE" ->
"+URFIVE", "+URSIX" -> "+URSIX", "+URSEVEN" -> "+URSEVEN", "+UREIGHT"
-> "+UREIGHT", "+URNINE" -> "+URNINE", "+HIABRIVATION" -> "+URSTBD",
"+HIDECIMAL" -> "+URDECIMAL", "+HITHSEP" -> "+URTHSEP", "+HIZERO" ->
"+URZERO", "+HIONE" -> "+URONE", "+HITWO" -> "+URTWO", "+HITHREE" ->
"+URTHREE", "+HIFOUR" -> "+URFOUR", "+HIFIVE" -> "+URFIVE", "+HISIX" -
> "+URSIX", "+HISEVEN" -> "+URSEVEN", "+HIEIGHT" -> "+UREIGHT",
"+HININE" -> "+URNINE"];
compose net
```

Seraiki/Devanagari to UIT Finite-state Transducer

```
clear stack
set char-encoding UTF-8
!*************************************************
! START
!*************************************************
! ******************
! Definition of vairables
! 44 consonants कखगघङचछजझञटठडढणतथदधनपफबभमयरलवशषसहक़ख़ग़ज़ड़ढ़फ़
ग़ज़ड़ब़
define CONSONANTS
[क|ख|ग|घ|ङ|च|छ|ज|झ|ञ|ट|ठ|ड|ढ|ण|त|थ|द|ध|न|प|फ|ब|भ|म|य|र|ल|व|श|ष|
स|ह|क़|ख़|ग़|ज़|ड़|ढ़|फ़|ग़|ज़|ड़|ब़];
define HINUMBER [०|१|२|३|४|५|६|७|८|९];
define ENGNUMBER [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9];
define HINUMBEREXPRESSION [ [HINUMBER [HINUMBER]*] ];
define ENGNUMBEREXPRESSION [ [ENGNUMBER [ENGNUMBER]*] ];
!******************
! Rules for Hindi to UIT Conversion
!******************
! ******************
! Other Symbols
```

```
read regex [ॅ -> ".", ः -> H];
!*******************
! Rules for Consonants
!*******************
! Simple consonants
read regex [क -> k, ख -> [k "_" h], ग -> g, घ -> [g "_" h], ङ -> [N 1], च
-> [t "_" S], छ -> [t "_" S "_" h], ज -> [d "_" Z], झ -> [d "_" Z "_"
h], ञ -> [N 2], ट -> [t "`"], ठ -> [t "`" "_" h], ड -> [d "`"], ढ -> [d
"`" "_" h], ण -> [n "`"], त -> [t "_" d], थ -> [t "_" d "_" h], द -> [d
"_" d], ध -> [d "_" d "_" h], न -> n, प -> p, फ -> [p "_" h], ब -> b, भ ->
[b "_" h], म -> m, य -> j, र -> r, ल -> l, व -> v, श -> S, ष -> [S 1], स ->
s, ह -> h, क़ -> q, ख़ -> x, ग़ -> G, ज़ -> z, ड़ -> [r "`"], ढ़ -> [r "`" "_"
h], फ़ -> f, ग़ -> [g 1], ज़ -> [d "_" Z 1], ड़ -> [d "`" 1], ब़ -> [b 1]];
! Other Aspirated Consonants
read regex [[ॅ ह] -> ["_" h] || [र | ल | म | न | व] _ ];
! Special case for YA
read regex [य -> [j h] || _ [.#. | " "]];
!*******************
! Rules for Nasalization
!*******************
read regex [ँ -> "~", ं -> "~"];
!*******************
! Rules for Vowels
!*******************
read regex [ऍ -> [I 2 e 1], ऐ -> [I 2 e 1], ऑ -> [I 2 o 1], ओ -> [I 2 o
1]];
! Dependent vowel sings that are very rare in thier use
read regex [ ॅ -> [e 1], े -> [e 1], ॉ -> [o 1], ो -> [o 1]];
! Dependent vowel signs that are very common in their use
read regex [ा -> [A 1], ि -> [I 1], ी -> [i 1], ु -> [U 1], ू -> [u 1], ृ ->
[r 1 1], े -> [e 1], ै -> ["{" 1], ो -> [o 1], ौ -> [O 1]];
read regex [अ -> "?", आ -> ["?" A 1], इ -> [I 2], उ -> [I 2 U 1], ई -> [I
2 i 1], [ऊ] -> [I 2 u 1], ए -> [I 2 e 1], [ऐ] -> [I 2 "{" 1], ओ -> [I 2 o
1], [औ] -> [I 2 O 1], ऋ -> [r 1]];
!read regex [[ा अ] -> "?"];
! when vowel sign U comes after a consonant and before the independent
vowel signs at the end of a word
read regex [[ु] -> [u 1] || CONSONANTS _ [[आ | ई | ए | ँ | ं] [.#. | "
"]]];
read regex [आ -> [A 1], ओ -> [o 1] || [इ | ई | ओ | ा | ि | ी | ु | ू | ृ | े
| ै | ो | ौ | ँ | ं | ॅ | ॉ | ॅ | े | I] _ ];
read regex [इ -> I || [A | o | ा | ो | आ | ओ] _ [आ | ओ | ा | ो]];
! special string at the start of a word n equivalent in Urdu is BEH +
ALEF + LAM
! read regex [[ब ि अ ल] -> [b I 1 A 1 l] || [.#. | " "] _ ];
! vowels at the start of a word
```

```
read regex [अ -> "@", आ -> A, इ -> I, ई -> i, उ -> U, ऊ -> u, ऋ -> [r 1],
ए -> e, ऐ -> "{", ओ -> o, औ -> O, ऍ -> e, ऎ -> e, ऑ-> o, ऒ -> o || [.#. |
" "| ◌] _ ];
read regex [[◌ु] -> [u 1], [◌ि] -> [i 1] || CONSONANTS _ [.#. | " "] ];
!*******************
! Rules for specail strings
!*******************
read regex [[प ◌ु र] -> [p u 1 r], [◌ा ब ◌ा द] -> [A b A 1 d "_" d], [अ ल ◌् ल
◌ा ह] -> ["@" l "." l A 1 h] || [? - [" " | .#.]] _ [.#. | " "]];
! *******************
! special words
read regex [[अ ल ◌् ल ◌ा ह] -> ["@" l "." l A 1 h], ॐ -> [O m], [न] -> [n A
1], [य ◌े] -> [j h], [व ◌ो] -> [v h], [व ◌े] -> [v h] || [" " | .#.] _ [.#. |
" "]];
!*******************
! Rules for Numbers
!*******************
read regex [० -> "+HIZERO", १ -> "+HIONE", २ -> "+HITWO", ३ -> "+HITHREE",
४ -> "+HIFOUR", ५ -> "+HIFIVE", ६ -> "+HISIX", ७ -> "+HISEVEN", ८ ->
"+HIEIGHT", ९ -> "+HININE", । -> "+SSTBD", ॥ -> "+DSTBD", "॰" ->
"+HIABRIVATION"];
! Decimal separator
read regex ["+DOT" -> "+HIDECIMAL", "+COMMA" -> "+HITHSEP" || HINUMBE-
REXPRESSION _ HINUMBEREXPRESSION];
read regex ["0" -> "+ZERO", 1 -> "+ONE", 2 -> "+TWO", 3 -> "+THREE", 4
-> "+FOUR", 5 -> "+FIVE", 6 -> "+SIX", 7 -> "+SEVEN", 8 -> "+EIGHT", 9
-> "+NINE"];
read regex ["+DOT" -> ["+DECIMAL"], "+COMMA" -> "+THSEP"  || ENGNUMBE-
REXPRESSION _ ENGNUMBEREXPRESSION];
! *******************
! Marks
read regex ["." -> "+DOT", "?" -> "+QMARK", "," -> "+COMMA", ";" ->
"+SEMICOLON", ":" -> "+COLON", "%" -> "+PERCENT", "+NL" -> "+NL"];
! *******************
! Hindi text normalization rules
! Removing all dot below from the text as they have been incorrectly
or mistakenly put in the text.
read regex [◌़ -> 0];
! normalizing consonants with a dot below
read regex [[क ◌़] -> क़, [ख ◌़] -> ख़,  [ग ◌़] -> ग़, [ज ◌़] -> ज़, [ड ◌़] -> ड़,
[ढ ◌़] -> ढ़, [फ ◌़] -> फ़];
!*************************************************
! END
!*************************************************
compose net
```

UIT to Seraiki/Devanagari Finite-state Transducer

```
clear stack
set char-encoding UTF-8
!*******************
```

```
!Definition of Varibales
!*******************
define CONSONANTS [b | [b "_" h] | p | [p "_" h] | [t "_" d] | [t "_"
d "_" h] | [t "`"] | [t "`" "_" h] | [s 1] | [d "_" Z] | [d "_" Z "_"
h] | [t "_" S] | [t "_" S "_" h] | [h 1] | x | [d "_" d] | [d "_" d
"_" h] | [d "`"] | [d "`" "_" h] | [z 1] | r | [r "_" h] | [r "`"] |
[r "`" "_" h] | z | Z | s | S | [s 2] | [z 2] | [t "_" d 1] | [z 3] |
G | f | q | k | [k "_" h] | g | [g "_" h] | l | [l "_" h] | m | [m "_"
h] | n | [n "_" h] | v | [v "_" h] | h | j | [t "_" d 2] | H | [N 1] |
[N 2] | [n "`"] | [l "`"] | [b 1] | [d "_" Z 1] | [d "`" 1] | [g 1]];
define VOWELS ["@" | A | I | U | e | "{" | i | u | O | o | [A 1] | [A
2] | [A 3] | [I 1] | [I 2] | [U 1] | [e 1] | ["{" 1] | [i 1] | [u 1] |
[O 1] | [o 1] | [e 3] | [e 4] | ["{" 3] | ["{" 4] | [i 2] | [u 2]];
!*******************
! Multi-character symbols
!*******************
read regex ["+DOT" -> ".", "+QMARK" -> "?", "+DECIMAL" -> ".",
"+COMMA" -> ",", "+NL" -> "+NL", "+THSEP" -> ",", "+SEMICOLON" -> ";",
"+COLON" -> ":", "+PERCENT" -> "%", "+ZERO" -> "0", "+ONE" -> 1,
"+TWO" -> 2, "+THREE" -> 3, "+FOUR" -> 4, "+FIVE" -> 5, "+SIX" -> 6,
"+SEVEN" -> 7, "+EIGHT" -> 8, "+NINE" -> 9, "+DSTBD" -> ॥, "+SSTBD" ->
।, "+HIDECIMAL" -> ".", "+HITHSEP" -> ",", "+HIZERO" -> ०, "+HIONE" ->
१, "+HITWO" -> २, "+HITHREE" -> ३, "+HIFOUR" -> ४, "+HIFIVE" -> ५,
"+HISIX" -> ६, "+HISEVEN" -> ७, "+HIEIGHT" -> ८, "+HININE" -> ९,
"+HIABRIVATION" -> ॰];
!*******************
! Diacritics
!*******************
read regex ["." -> ◌़];
!*******************
! Consonants
!*******************
! Simple Consonants
read regex [b -> ब, p -> प, [t "_" d] -> त, [t "`"] -> ट, x  -> ख़, [d
"_" d] -> द, [d "`"] -> ड, r -> र, z -> ज़, Z -> ज़, s -> स, S -> श, G
-> ग़, f -> फ़, q -> क़, k -> क, g -> ग, l -> ल, m -> म, n -> न, v ->
व, h -> ह, j -> य, H -> ◌ः, [N 1] -> ङ, [N 2] -> ञ];
! Consonants Set 2
read regex [[s 1] -> स, [d "_" Z] -> ज, [t "_" S] -> च, [h 1] -> ह,
[z 1] -> ज़, [r "`"] -> ड़, [S 1] -> ष, [s 2] -> स, [z 2] -> ज़, [t "_"
d] -> त, [z 3] -> ज़, [t "_" d 2] -> त, [n "`"] -> ण, [l "`"] -> ल, [b
1] -> ब़, [d "`" 1] -> ड़, [g 1] -> ग़];
! Consonants Set 3
read regex [[d "_" Z 1] -> ज़];
! Aspirated Consonants
read regex [ ["_" h] -> [◌़ ह]];
read regex [[b "_" h] -> भ, [p "_" h] -> फ, [t "_" d "_" h] -> थ, [t
"`" "_" h] -> ठ, [d "_" Z "_" h] -> झ, [t "_" S "_" h] -> छ, [d "_" d
```

```
"_" h] -> ਧ, [d "`" "_" h] -> ਢ, [r "`" "_" h] -> ੜ, [k "_" h] -> ਖ,
[g "_" h] -> ਘ];
! Germinated Simple Consonants
!read regex [[b "." b] -> [ੱਬ], [p "." p] -> [ੱਪ], [t "_" d "." t "_"
d] -> [ੱਤ], [t "`" "." t "`"] -> [ੱਟ], [s 1 "." s 1] -> [ੱਸ], [d "_"
Z "." d "_" Z] -> [ੱਜ], [t "_" S "." t "_" S] -> [ੱਚ], [h 1 "." h 1]
-> [ੱਹ], [x "." x] -> [ੱਖ], [d "_" d "." d "_" d] -> [ੱਦ], [d "`" "."
d "`"] -> [ੱਡ], [z 1 "." z 1] -> [ੱਜ਼], [r "." r] -> [ੱਰ], [r "`" "."
r "`"] -> [ੱੜ], [z "." z] -> [ੱਜ਼], [Z "." Z] -> [ੱਜ਼], [s "." s] -> [ੱ
ਸ], [S "." S] -> [ੱਸ਼], [S 1 "." S 1] -> [ੱਸ਼], [s 2 "." s 2] -> [ੱਸ],
[z 2 "." z 2] -> [ੱਜ਼], [t "_" d 1 "." t "_" d 1] -> [ੱਤ], [z 3 "." z
3] -> [ੱਜ਼], [G "." G] -> [ੱਗ਼], [f "." f] -> [ੱਫ਼], [q "." q] -> [ੱ
ਕ਼], [k "." k] -> [ੱਕ], [g "." g] -> [ੱਗ], [l "." l] -> [ੱਲ], [m "."
m] -> [ੱਮ], [n "." n] -> [ੱਨ], [v "." v] -> [ੱਵ], [h "." h] -> [ੱਹ],
[j "." j] -> [ੱਯ], [t "_" d 2 "." t "_" d 2] -> [ੱਤ], [n "`" "." n
"`"] -> [ੱਣ], [l "`" "." l "`"] -> [ੱਲ਼], [b 1 "." b 1] -> [ੱਬ], [d
"_" Z 1 "." d "_" Z 1] -> [ੱਜ], [d "`" 1 "." d "`" 1] -> [ੱਡ], [g 1
"." g 1] -> [ੱਗ], [N 1 "." N 1] -> [ੱਙ], [N 2 "." N 2] -> [ੱਞ]];
! Germinated Aspirated Consonants
!read regex [[b "." b "_" h] -> [ੱਭ], [p "." p "_" h] -> [ੱਫ], [t "_"
d "." t "_" d "_" h] -> [ੱਥ], [t "`" "." t "`" "_" h] -> [ੱਠ], [d "_"
Z "." d "_" Z "_" h] -> [ੱਝ], [t "_" S "." t "_" S "_" h] -> [ੱਛ], [d
"_" d "." d "_" d "_" h] -> [ੱਧ], [d "`" "." d "`" "_" h] -> [ੱਢ], [k
"." k "_" h] -> [ੱਖ], [g "." g "_" h] -> [ੱਘ]];
! ******************
! Nazalization
read regex ["~" -> ◌ं];
read regex ["~" -> ◌ं || [CONSONANTS | ASPICONSONANTS] _ ];
read regex [ ["~" 1] -> ◌ं];
!******************
! Vowels
!******************
! Vowel characters
read regex [A -> ਆ, "@" -> ਅ, I -> �इ, U -> ਉ, i -> ਈ, o -> ਓ, O ->
ਔ, u -> ਊ, e -> ਏ, "{" -> ਐ];
read regex [[e 3] -> ਏ, ["{" 3] -> ਐ];
! Vowel signs
read regex [[A 1] -> ◌ਾ, [I 1] -> ◌ਿ, [U 1] -> ◌ੁ, [i 1] -> ◌ੀ, [o 1] ->
◌ੋ, [O 1] -> ◌ੌ, [u 1] -> ◌ੂ, [e 1] -> ◌ੇ, ["{" 1] -> ◌ੈ, [e 4] -> ◌ੇ, ["{"
4] -> ◌ੈ];
```

```
! ******************
! Special cases
! ******************
! Special cases of vowels
read regex [["@" n] -> न];
read regex [[A 2] -> ा, [A 3] -> ा, [I 2] -> इ, [i 2] -> ी, [u 2] -
> ू, ["@" n 1] -> न, [I 1 n 1] -> [ि न], [U 1 n 1] -> [ु न]];
read regex [[u 1] -> ु || CONSONANTS _ [[A | i | e | "~"] .#.]];
! Vowel with Ain
read regex ["?" -> अ];
read regex ["?" -> ा || CONSONANTS _ ];
read regex [["?" A 1] -> आ, A 1 "?" -> [ा इ]];
read regex [["@" "?"] -> ए, ["?" I 1] -> इ, ["?" U 1] -> उ, ["?" e 1]
-> ए, ["?" i 1] -> ई, ["?" "{" 1] -> ऐ, ["?" o 1] -> ओ, ["?" O 1] ->
औ, ["?" u 1] -> ऊ, ["?" e 4] -> ए, ["?" "{" 4] -> ऐ];
!read regex [["@" "?"] -> ए, ["?" I 1] -> इ, ["?" U 1] -> उ, ["?" e 1]
-> ए, ["?" i 1] -> ई, ["?" "{" 1] -> ऐ, ["?" o 1] -> ओ, ["?" O 1] ->
औ, ["?" u 1] -> ऊ, ["?" e 4] -> ए, ["?" "{" 4] -> ऐ || [.#. | " " |
"."] _ ];
! Ain after the final i sound in the word
read regex [["?" i 1] -> [ा ई] || _ [.#. | " "]];
read regex [[I 2 i 1] -> ई, [I 2 e 1] -> ए, [I 2 "{" 1] -> ऐ, [I 2 o
1] -> ओ, [I 2 u 1] -> ऊ, [I 2 U 1] -> उ, [I 2 O 1] -> औ, [I 2 e 4] -
> ए];
read regex [[A 1] -> आ, [I 1] -> इ, [U 1] -> उ, [i 1] -> ई, [o 1] ->
ओ, [O 1] -> औ, [u 1] -> ऊ, [e 1] -> ए, ["{" 1] -> ऐ, [e 4] -> ए,
["{" 4] -> ऐ, [r 1 1] -> ऋ || VOWELS _ ];
! ******************
! Compound Words
read regex [["-" e "-"] -> ["-" ए "-"], ["-" e 1 "-"] -> ["-" ए "-"] || _
[" "]];
!******************
! Special words
read regex [ ["@" l "." l A 1 h] -> [अ ल ् ल ा ह], [p u 1 r] -> [प ु र]
|| _ [.#. | " "]];
read regex [["@" l "." l A 1 h] -> [अ ल ् ल ा ह], [v h] -> [व ो], [j h]
-> [य े], [n A 1] -> न, [O m] -> ॐ || [.#. | " " ] _ [ .#. | " " ]];
! ******************
! Multi-character symbols
read regex ["+DOT" -> "+DOT", "+QMARK" -> "+QMARK", "+DECIMAL" ->
"+DECIMAL", "+COMMA" -> "+COMMA", "+NL" -> "+NL", "+THSEP" ->
"+THSEP", "+SEMICOLON" -> "+SEMICOLON", "+COLON" -> "+COLON",
"+PERCENT" -> "+PERCENT", "+ZERO" -> "+ZERO", "+ONE" -> "+ONE", "+TWO"
-> "+TWO", "+THREE" -> "+THREE", "+FOUR" -> "+FOUR", "+FIVE" ->
"+FIVE", "+SIX" -> "+SIX", "+SEVEN" -> "+SEVEN", "+EIGHT" -> "+EIGHT",
"+NINE" -> "+NINE", "+DSTBD" -> "+DSTBD", "+SSTBD" -> "+SSTBD",
```

```
"+GURDECIMAL" -> "+HIDECIMAL", "+GURTHSEP" -> "+HITHSEP", "+GURZERO" -
> "+HIZERO", "+GURONE" -> "+HIONE", "+GURTWO" -> "+HITWO", "+GURTHREE"
-> "+HITHREE", "+GURFOUR" -> "+HIFOUR", "+GURFIVE" -> "+HIFIVE",
"+GURSIX" -> "+HISIX", "+GURSEVEN" -> "+HISEVEN", "+GUREIGHT" ->
"+HIEIGHT", "+GURNINE" -> "+HININE", "+URQMARK" -> "+QMARK", "+URSTBD"
-> "+SSTBD", "+URCOMMA" -> "+COMMA", "+URPERCENT" -> "+PERCENT",
"+URSEMICOLON" -> "+SEMICOLON", "+URSTAR" -> "*", "+URDECIMAL" ->
"+HIDECIMAL", "+URTHSEP" -> "+HITHSEP", "+URZERO" -> "+HIZERO",
"+URONE" -> "+HIONE", "+URTWO" -> "+HITWO", "+URTHREE" -> "+HITHREE",
"+URFOUR" -> "+HIFOUR", "+URFIVE" -> "+HIFIVE", "+URSIX" -> "+HISIX",
"+URSEVEN" -> "+HISEVEN", "+UREIGHT" -> "+HIEIGHT", "+URNINE" ->
"+HININE", "+HIABRIVATION" -> "+HIABRIVATION", "+HIDECIMAL" ->
"+HIDECIMAL", "+HITHSEP" -> "+HITHSEP", "+HIZERO" -> "+HIZERO",
"+HIONE" -> "+HIONE", "+HITWO" -> "+HITWO", "+HITHREE" -> "+HITHREE",
"+HIFOUR" -> "+HIFOUR", "+HIFIVE" -> "+HIFIVE", "+HISIX" -> "+HISIX",
"+HISEVEN" -> "+HISEVEN", "+HIEIGHT" -> "+HIEIGHT", "+HININE" ->
"+HININE"];
compose net
```

Word Ambiguity Transducer

```
clear stack
set char-encoding UTF-8
define CONSONANTS [क | ख | ग | घ | ङ | च | छ | ज | झ | ञ | ट | ठ | ड
| ढ | ण | त | थ | द | ध | न | प | फ | ब | भ | म | य | र | ल | व | श
| ष | स | ह | क़ | ख़ | ग़ | ज़ | ड़ | ढ़ | फ़];
define VOWELSING [ा | ि | ी | ु | ू | ृ | ॅ | े | े | ै | ॉ | ो | ो
| ौ | ् ];
read regex [[क क] (->) क, [क ख] (->) ख, [ग ग] (->) ग, [ग घ] (->) घ, [च
च] (->) च, [च छ] (->) छ, [ज ज] (->) ज, [ज झ] (->) झ, [ट ट] (->) ट, [ट ठ]
(->) ठ, [ड ड] (->) ड, [ड ढ] (->) ढ, [ण ण] (->) ण, [त त] (->) त, [त थ] (->)
थ, [द द] (->) द, [द ध] (->) ध, [न न] (->) न, [प प] (->) प, [प फ] (->) फ,
[ब ब] (->) ब, [ब भ] (->) भ, [म म] (->) म, [य य] (->) य, [र र] (->) र, [ल ल]
(->) ल, [व व] (->) व, [श श] (->) श, [ष ष] (->) ष, [स स] (->) स, [ह ह] (->)
ह, [क़ क़] (->) क़, [ख़ ख़] (->) ख़, [ग़ ग़] (->) ग़, [ज़ ज़] (->) ज़, [ड़ ड़] (->) ड़,
[ड़ ढ़] (->) ढ़, [फ़ फ़] (->) फ़];
read regex [् -> 0 || _ [.#.| अ | आ | इ | ई | उ | ऊ | ऋ | ए | ऐ | ओ
| औ |ा | ि | ी | ु | ू | ृ | ॅ | े | े | ै | ॉ | ो | ो | ौ | ् ]];
read regex [् -> 0 || [.#.| अ | आ | इ | ई | उ | ऊ | ऋ | ए | ऐ | ओ |
औ |ा | ि | ी | ु | ू | ृ | ॅ | े | े | ै | ॉ | ो | ो | ौ | ् ] _ ];
read regex [् -> 0 || [? - [र | ल | म | न]] _ [ह]];
read regex [् -> 0 || [[र | ल | म | न]] _ [? - ह]];
read regex [् -> 0 || [? - [र | ल | म | न]] _ [? - ह]];
read regex [न (->) ं || [? - [.#.|ा]] _ [? - [.#.|ा | ि | ी | ु | ू
| ृ | ॅ | े | े | ै | ॉ | ो | ो | ौ | ् ]]];
```

```
read regex [ङ -> [ं ग], ञ -> [ं ज] || [? - .#.] _ ];
read regex [ङ -> [न ग], ञ -> [न ज] || .#. _ ];
read regex [क (->) क़, ख (->) ख़, ग (->) ग़, ज (->) ज़, ड (->) ड़, ढ (->) ढ़,
फ (->) फ़];
read regex [ै (->) े, ी (->) े, ू (->) ो, ौ (->) ो, ि (->) 0, ु (->) 0
|| [CONSONANTS] _ ];
read regex [ी (->) े || [CONSONANTS] _ [? - .#.]];
read regex [ि -> ी, ु -> ो, ु -> ू || [CONSONANTS] _ [.#. | [ं .#.]]];
!read regex [0 (->) अ || [ू | ु] _ .#.];
read regex [ा (->) अ || [CONSONANTS] _ [? - .#.]];
read regex [य (->) े || [CONSONANTS] _ [? - [ा | ि | ी | ु | ू | ृ
| ॅ | ॆ | े | ै | ॉ | ॊ | ो | ौ | ् | .#.]]];
read regex [व (->) ो || [CONSONANTS] _ [? - [ा | ि | ी | ु | ू | ृ
| ॅ | ॆ | े | ै | ॉ | ॊ | ो | ौ | ् ]]];
! अ आ इ ई उ ऊ ऋ ए ऐ ओ औ
read regex [इ (->) अ, ई (->) ए, ऐ (->) ए, ऊ (->) ओ, औ (->) ओ || [.#.
| ्] _ ];
read regex [ः -> ह, ँ -> ं, ृ -> र, ॅ -> े, ॆ -> े, ॉ -> ो, ॊ -> ो, ॠ
-> र, ऍ -> ए, ऎ -> ए, ऑ -> ओ, ऒ -> ओ, ण -> न, ष -> श];
compose net
```

Annex 7 Results

Hindi to Urdu Results

Character alignment results

Models	Sentence Accuracy		Word accuracy		Character accuracy	
	default output	processed output	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering			59.6%	74.9%	94.1%	94.8%
Hi-UrWD-UWLMWD-No-Reordering			59.7%	75.3%	94.3%	95.0%
Hi-UrWD-UWLMWD-Tuned-With-Reordering			69.9%	79.6%	95.4%	95.9%
Hi-UrWD-UWLMWD-Tuned-No-Reordering			69.9%	**79.7%**	95.5%	95.9%
Hi-UrWD-USLM-With-Reordering			16.1%	48.7%	74.8%	89.1%
Hi-UrWD-USLM-No-Reordering			17.5%	53.4%	77.4%	92.1%
Hi-UrWD-USLM-Tuned-With-Reordering			57.4%	76.0%	92.9%	95.0%
Hi-UrWD-USLM-Tuned-No-Reordering			57.6%	76.6%	92.9%	95.0%
Hi-UrWD-USLM+UWLMWD-With-Reordering			35.4%	67.8%	85.2%	93.7%
Hi-UrWD-USLM+UWLMWD -No-Reordering			36.1%	69.9%	86.3%	94.7%
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering			**70.0%**	**79.7%**	95.4%	95.9%
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering			**70.0%**	**79.7%**	95.4%	95.9%
Hi-UrWOD-UWLMWOD-With-Reordering			59.4%	59.4%	94.7%	94.7%
HiWOD-Ur-UWLMWOD-No-Reordering			59.5%	59.5%	94.8%	94.8%
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering			77.5%	77.5%	95.2%	95.2%
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering			77.5%	77.5%	95.2%	95.2%
Hi-UrWOD-USLM-With-Reordering			48.0%	48.0%	90.4%	90.4%
Hi-UrWOD-USLM-No-Reordering			50.4%	50.4%	91.7%	91.7%
Hi-UrWOD-USLM-Tuned-With-Reordering			77.8%	77.8%	95.3%	95.3%
Hi-UrWOD-USLM-Tuned-No-Reordering			77.9%	77.9%	95.3%	95.3%
Hi-UrWOD-USLM+UWLMWOD-With-Reordering			65.1%	65.1%	93.6%	93.6%
Hi-UrWOD-USLM+UWLMWOD -No-Reordering			65.7%	65.7%	93.9%	93.9%
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering			**78.3%**	**78.3%**	95.3%	95.3%
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering			**78.3%**	**78.3%**	95.3%	95.3%

Table 83: HU Test Set 1 results of Hindi to Urdu SMT systems (character alignments)

Models	Sentence Accuracy		Word accuracy		Character accuracy	
	default output	processed output	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering	0	2.2%	24.3%	64.1%	89.5%	91.1%
Hi-UrWD-UWLMWD-No-Reordering	0	2.2%	24.3%	64.2%	89.5%	91.1%
Hi-UrWD-UWLMWD-Tuned-With-Reordering	0	1.3%	36.8%	60.6%	88.9%	90.3%
Hi-UrWD-UWLMWD-Tuned-No-Reordering	0	1.3%	36.8%	60.6%	88.9%	90.3%
Hi-UrWD-USLM-With-Reordering	1.3%	2.2%	52.7%	67.0%	87.7%	92.7%
Hi-UrWD-USLM-No-Reordering	1.3%	2.2%	52.8%	67.8%	88.2%	93.1%
Hi-UrWD-USLM-Tuned-With-Reordering	0.9%	1.3%	37.9%	61.9%	89.3%	90.8%
Hi-UrWD-USLM-Tuned-No-Reordering	0.9%	1.3%	37.9%	61.9%	89.3%	90.8%
Hi-UrWD-USLM+UWLMWD-With-Reordering	0.9%	4%	57.3%	70.9%	90.5%	93.2%
Hi-UrWD-USLM+UWLMWD -No-Reordering	0.9%	4%	57.3%	71.1%	90.6%	93.3%
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering	0.4%	1.3%	36.4%	60.4%	88.8%	90.2%
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering	0.4%	1.3%	36.4%	60.4%	88.8%	90.2%
Hi-UrWOD-UWLMWOD-With-Reordering	1.8%	1.8%	62.9%	62.9%	91.4%	91.4%
HiWOD-Ur-UWLMWOD-No-Reordering	1.8%	1.8%	62.9%	62.9%	91.4%	91.4%
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering	1.3%	1.3%	62.0%	62.0%	90.5%	90.5%
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering	1.3%	1.3%	62.0%	62.0%	90.5%	90.5%
Hi-UrWOD-USLM-With-Reordering	2.2%	2.2%	67.2%	67.2%	93.7%	93.7%
Hi-UrWOD-USLM-No-Reordering	2.7%	2.7%	67.2%	67.2%	93.8%	93.8%
Hi-UrWOD-USLM-Tuned-With-Reordering	1.3%	1.3%	64.5%	64.5%	91.2%	91.2%
Hi-UrWOD-USLM-Tuned-No-Reordering	1.3%	1.3%	64.5%	64.5%	91.2%	91.2%
Hi-UrWOD-USLM+UWLMWOD-With-Reordering	4%	4%	68.3%	68.3%	92.9%	92.9%
Hi-UrWOD-USLM+UWLMWOD -No-Reordering	4%	4%	68.3%	68.3%	92.9%	92.9%
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering	1.3%	1.3%	63.4%	63.4%	90.9%	90.9%
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering	1.3%	1.3%	63.4%	63.4%	90.9%	90.9%

Table 84: HU Test Set 3 results of Hindi to Urdu SMT systems (character alignments)

Cluster alignment results

Models	Sentence Accuracy		Word accuracy		Character accuracy	
	default output	processed output	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering			59.6%	74.9%	93.7%	94.6%
Hi-UrWD-UWLMWD-No-Reordering			59.7%	75.3%	93.8%	94.7%
Hi-UrWD-UWLMWD-Tuned-With-Reordering			69.9%	79.6%	94.6%	95.5%
Hi-UrWD-UWLMWD-Tuned-No-Reordering			69.9%	79.7%	94.6%	95.5%
Hi-UrWD-USLM-With-Reordering			16.1%	48.7%	74.1%	89.1%
Hi-UrWD-USLM-No-Reordering			17.5%	53.4%	76.3%	91.6%
Hi-UrWD-USLM-Tuned-With-Reordering			57.4%	76.0%	91.8%	94.3%
Hi-UrWD-USLM-Tuned-No-Reordering			57.6%	76.6%	92.0%	94.5%
Hi-UrWD-USLM+UWLMWD-With-Reordering			35.4%	67.8%	82.6%	93.3%
Hi-UrWD-USLM+UWLMWD -No-Reordering			36.1%	69.9%	83.5%	94.2%
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering			70.0%	79.7%	94.6%	95.5%
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering			70.0%	79.7%	94.6%	95.5%
Hi-UrWOD-UWLMWOD-With-Reordering			61.9%	61.9%	94.5%	94.5%
HiWOD-Ur-UWLMWOD-No-Reordering			62.1%	62.1%	94.6%	94.6%
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering			78.1%	78.1%	94.9%	94.9%
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering			78.1%	78.1%	94.9%	94.9%
Hi-UrWOD-USLM-With-Reordering			50.5%	50.5%	90.3%	90.3%
Hi-UrWOD-USLM-No-Reordering			53.7%	53.7%	92.0%	92.0%
Hi-UrWOD-USLM-Tuned-With-Reordering			78.2%	78.2%	94.9%	94.9%
Hi-UrWOD-USLM-Tuned-No-Reordering			78.3%	78.3%	95.0%	95.0%
Hi-UrWOD-USLM+UWLMWOD-With-Reordering			67.1%	67.1%	93.6%	93.6%
Hi-UrWOD-USLM+UWLMWOD -No-Reordering			68.1%	68.1%	94.1%	94.1%
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering			78.4%	78.4%	95.0%	95.0%
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering			78.4%	78.4%	95.0%	95.0%

Table 85: HU Test Set 1 results of Hindi to Urdu SMT systems (cluster alignments)

Models	Sentence Accuracy		Word accuracy		Character accuracy	
	default output	processed output	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering	0.5%	3%	37.3%	64.8%	87.4%	92.8%
Hi-UrWD-UWLMWD-No-Reordering	0.5%	3%	37.3%	64.8%	87.4%	92.8%
Hi-UrWD-UWLMWD-Tuned-With-Reordering	1%	3%	37.9%	66.2%	87.9%	93.2%
Hi-UrWD-UWLMWD-Tuned-No-Reordering	1%	3%	37.9%	66.2%	87.9%	93.2%
Hi-UrWD-USLM-With-Reordering	1%	4.5%	52.7%	65.2%	85.2%	92.6%
Hi-UrWD-USLM-No-Reordering	1%	5.5%	53.4%	66.6%	86.2%	93.6%
Hi-UrWD-USLM-Tuned-With-Reordering	0.5%	3%	32.7%	61.9%	85.7%	91%
Hi-UrWD-USLM-Tuned-No-Reordering	0.5%	4.5%	33.0%	62.3%	85.8%	91.1%
Hi-UrWD-USLM+UWLMWD-With-Reordering	1%	5%	51.6%	69.0%	87.6%	93.9%
Hi-UrWD-USLM+UWLMWD -No-Reordering	1%	5%	51.6%	69.2%	87.7%	94%
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering	1%	3%	37.7%	66.5%	88%	93.4%
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering	1%	3%	37.7%	66.5%	88%	93.4%
Hi-UrWOD-UWLMWOD-With-Reordering	3%	3%	64.3%	64.3%	93.1%	93.1%
HiWOD-Ur-UWLMWOD-No-Reordering	3%	3%	64.3%	64.3%	93.1%	93.1%
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering	3%	3%	64.7%	64.7%	92.2%	92.2%
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering	3%	3%	64.7%	64.7%	92.2%	92.2%
Hi-UrWOD-USLM-With-Reordering	5.5%%	5.5%%	65.3%	65.3%	93%	93%
Hi-UrWOD-USLM-No-Reordering	5.5%%	5.5%%	66.2%	66.2%	93.6%	93.6%
Hi-UrWOD-USLM-Tuned-With-Reordering	3.5%	3.5%	65.2%	65.2%	92.3%	92.3%
Hi-UrWOD-USLM-Tuned-No-Reordering	3.5%	3.5%	65.4%	65.4%	92.3%	92.3%
Hi-UrWOD-USLM+UWLMWOD-With-Reordering	3.5%	3.5%	69.5%	69.5%	93.6%	93.6%
Hi-UrWOD-USLM+UWLMWOD -No-Reordering	3.5%	3.5%	69.7%	69.7%	93.6%	93.6%
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering	3%	3%	65.3%	65.3%	92.3%	92.3%
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering	3%	3%	65.3%	65.3%	92.3%	92.3%

Table 86: HU Test Set 2 results of Hindi to Urdu SMT systems (cluster alignments)

Models	Sentence Accuracy		Word accuracy		Character accuracy	
	default output	processed output	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering	0.4%	0.9%	36.7%	61.9%	87.5%	90.4%
Hi-UrWD-UWLMWD-No-Reordering	0.4%	0.9%	36.7%	61.9%	87.5%	90.4%
Hi-UrWD-UWLMWD-Tuned-With-Reordering	0.4%	1.3%	38.0%	62.2%	87.3%	90.4%
Hi-UrWD-UWLMWD-Tuned-No-Reordering	0.4%	1.3%	38.0%	62.2%	87.3%	90.3%
Hi-UrWD-USLM-With-Reordering	0.9%	4.4%	57.8%	69.0%	88.7%	93.2%
Hi-UrWD-USLM-No-Reordering	0.9%	4.9%	58.0%	69.3%	89%	93.4%
Hi-UrWD-USLM-Tuned-With-Reordering	0.4%	0.9%	34.9%	59.8%	85.8%	88.8%
Hi-UrWD-USLM-Tuned-No-Reordering	0.4%	0.9%	35.1%	60.0%	85.9%	88.9%
Hi-UrWD-USLM+UWLMWD-With-Reordering	0.9%	3.1%	56.3%	67.2%	88.6%	92.1%
Hi-UrWD-USLM+UWLMWD -No-Reordering	0.9%	3.1%	56.3%	67.3%	88.7%	92.3%
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering	0.4%	1.3%	37.9%	62.6%	87.4%	90.5%
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering	0.4%	1.3%	37.9%	62.5%	87.4%	90.5%
Hi-UrWOD-UWLMWOD-With-Reordering	1.8%	1.8%	63.2%	63.2%	91.3%	91.3%
HiWOD-Ur-UWLMWOD-No-Reordering	1.8%	1.8%	63.2%	63.2%	91.3%	91.3%
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering	1.3%	1.3%	62.4%	62.4%	90.3%	90.3%
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering	1.3%	1.3%	62.4%	62.4%	90.3%	90.3%
Hi-UrWOD-USLM-With-Reordering	2.2%	2.2%	67.3%	67.3%	93.5%	93.5%
Hi-UrWOD-USLM-No-Reordering	2.7%	2.7%	67.4%	67.4%	93.6%	93.6%
Hi-UrWOD-USLM-Tuned-With-Reordering	1.3%	1.3%	63.1%	63.1%	90.5%	90.5%
Hi-UrWOD-USLM-Tuned-No-Reordering	1.3%	1.3%	63.1%	63.1%	90.5%	90.5%
Hi-UrWOD-USLM+UWLMWOD-With-Reordering	3.5%	3.5%	68.0%	68.0%	92.7%	92.7%
Hi-UrWOD-USLM+UWLMWOD -No-Reordering	3.5%	3.5%	68.0%	68.0%	92.7%	92.7%
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering	1.3%	1.3%	63.4%	63.4%	90.6%	90.6%
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering	1.3%	1.3%	63.4%	63.4%	90.6%	90.6%

Table 87: HU Test Set 3 results of Hindi to Urdu SMT systems (cluster alignments)

BLEU and NIST scores

In this section, we will give n-gram measures for our Hindi to Urdu SMT systems.

Models	NIST Score		BLEU Score	
	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering	5.9784	8.0052	0	0
Hi-UrWD-UWLMWD-No-Reordering	6.0158	8.0509	0	0
Hi-UrWD-UWLMWD-Tuned-With-Reordering	7.8037	8.8162	0	0
Hi-UrWD-UWLMWD-Tuned-No-Reordering	7.8079	8.8287	0	0
Hi-UrWD-USLM-With-Reordering	1.5946	5.1224	0	0
Hi-UrWD-USLM-No-Reordering	1.7343	5.7940	0	0
Hi-UrWD-USLM-Tuned-With-Reordering	6.4067	8.4455	0	0
Hi-UrWD-USLM-Tuned-No-Reordering	6.4067	8.4496	0	0
Hi-UrWD-USLM+UWLMWD-With-Reordering	4.2943	7.4975	0	0
Hi-UrWD-USLM+UWLMWD -No-Reordering	4.3898	7.7600	0	0
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering	7.8189	8.8404	0	0
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering	7.8230	8.8528	0	0
Hi-UrWOD-UWLMWOD-With-Reordering	1.1582	6.5470	0	0
HiWOD-Ur-UWLMWOD-No-Reordering	1.1582	6.5632	0	0
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering	1.2787	8.5562	0	0
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering	1.2787	8.5562	0	0
Hi-UrWOD-USLM-With-Reordering	0.7026	5.2779	0	0
Hi-UrWOD-USLM-No-Reordering	0.7317	5.5465	0	0
Hi-UrWOD-USLM-Tuned-With-Reordering	1.2832	8.5924	0	0
Hi-UrWOD-USLM-Tuned-No-Reordering	1.2832	8.5966	0	0
Hi-UrWOD-USLM+UWLMWOD-With-Reordering	1.1016	7.1791	0	0
Hi-UrWOD-USLM+UWLMWOD -No-Reordering	1.1016	7.2399	0	0
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering	1.2874	8.6457	0	0
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering	1.2874	8.6457	0	0

Table 88: HU Test Set 1 results of Hindi toUrdu SMT System (Character Alignments)

Models	NIST Score		BLEU Score	
	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering	6.5945	8.1599	0	0
Hi-UrWD-UWLMWD-No-Reordering	6.6109	8.1964	0	0
Hi-UrWD-UWLMWD-Tuned-With-Reordering	7.7555	8.6802	0	0
Hi-UrWD-UWLMWD-Tuned-No-Reordering	7.7555	8.6843	0	0
Hi-UrWD-USLM-With-Reordering	1.7959	5.2422	0	0
Hi-UrWD-USLM-No-Reordering	1.9101	5.7714	0	0
Hi-UrWD-USLM-Tuned-With-Reordering	6.4113	8.3427	0	0
Hi-UrWD-USLM-Tuned-No-Reordering	6.4278	8.4063	0	0
Hi-UrWD-USLM+UWLMWD-With-Reordering	3.8957	7.3650	0	0
Hi-UrWD-USLM+UWLMWD -No-Reordering	3.9800	7.6120	0	0
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering	7.7585	8.6825	0	0
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering	7.7585	8.6825	0	0
Hi-UrWOD-UWLMWOD-With-Reordering	1.1748	6.8304	0	0
HiWOD-Ur-UWLMWOD-No-Reordering	1.1748	6.8511	0	0
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering	1.2791	8.6160	0	0
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering	1.2791	8.6160	0	0
Hi-UrWOD-USLM-With-Reordering	0.7617	5.5584	0	0
Hi-UrWOD-USLM-No-Reordering	0.7991	5.9138	0	0
Hi-UrWOD-USLM-Tuned-With-Reordering	1.2753	8.6350	0	0
Hi-UrWOD-USLM-Tuned-No-Reordering	1.2711	8.6384	0	0
Hi-UrWOD-USLM+UWLMWOD-With-Reordering	1.1265	7.4028	0	0
Hi-UrWOD-USLM+UWLMWOD -No-Reordering	1.1348	7.5124	0	0
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering	1.2749	8.6548	0	0
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering	1.2708	8.6506	0	0

Table 89: HU Test Set 1 results of Hindi toUrdu SMT System (Cluster Alignments)

Models	NIST Score		BLEU Score	
	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering	2.2444	6.8904	0.0299	0.3465
Hi-UrWD-UWLMWD-No-Reordering	2.2444	6.8904	0.0299	0.3465
Hi-UrWD-UWLMWD-Tuned-With-Reordering	3.0665	6.7243	0.0659	0.3173
Hi-UrWD-UWLMWD-Tuned-No-Reordering	3.0665	6.7260	0.0659	0.3177
Hi-UrWD-USLM-With-Reordering	4.6298	6.3587	0.1590	0.2930
Hi-UrWD-USLM-No-Reordering	4.7342	6.6469	0.1644	0.3167
Hi-UrWD-USLM-Tuned-With-Reordering	3.0698	6.7467	0.0631	0.3150
Hi-UrWD-USLM-Tuned-No-Reordering	3.0698	6.7439	0.0631	0.3149
Hi-UrWD-USLM+UWLMWD-With-Reordering	4.7808	6.5498	0.1875	0.4221
Hi-UrWD-USLM+UWLMWD -No-Reordering	4.8110	7.6274	0.1889	0.4291
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering	3.0062	6.6902	0.0627	0.3141
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering	3.0062	6.6930	0.0627	0.3143
Hi-UrWOD-UWLMWOD-With-Reordering	3.6281	6.5377	0.0942	0.3177
HiWOD-Ur-UWLMWOD-No-Reordering	3.6281	6.5377	0.0942	0.3177
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering	3.5007	6.9538	0.0834	0.3400
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering	3.5007	6.9538	0.0834	0.3400
Hi-UrWOD-USLM-With-Reordering	3.5395	6.5059	0.0884	0.3031
Hi-UrWOD-USLM-No-Reordering	3.5677	6.5928	0.0897	0.3098
Hi-UrWOD-USLM-Tuned-With-Reordering	3.5259	6.9494	0.0842	0.3357
Hi-UrWOD-USLM-Tuned-No-Reordering	3.5338	6.9578	0.0843	0.3365
Hi-UrWOD-USLM+UWLMWOD-With-Reordering	3.6108	7.3188	0.0934	0.3876
Hi-UrWOD-USLM+UWLMWOD -No-Reordering	3.6180	7.3288	0.0935	0.3882
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering	3.4818	6.9491	0.0832	0.3389
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering	3.4897	6.9575	0.0833	0.3398

Table 90: HU Test Set 2 results of Hindi toUrdu SMT System (Character Alignments)

Models	NIST Score		BLEU Score	
	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering	3.0900	6.6153	0.0659	0.3235
Hi-UrWD-UWLMWD-No-Reordering	3.0900	6.6169	0.0659	0.3243
Hi-UrWD-UWLMWD-Tuned-With-Reordering	3.1825	6.8067	0.0713	0.3368
Hi-UrWD-UWLMWD-Tuned-No-Reordering	3.1825	6.8006	0.0713	0.3364
Hi-UrWD-USLM-With-Reordering	4.8674	6.5294	0.1978	0.3310
Hi-UrWD-USLM-No-Reordering	4.9680	6.7265	0.2058	0.3495
Hi-UrWD-USLM-Tuned-With-Reordering	2.8450	6.3987	0.0564	0.2885
Hi-UrWD-USLM-Tuned-No-Reordering	2.8870	6.4584	0.0591	0.2963
Hi-UrWD-USLM+UWLMWD-With-Reordering	4.8079	7.1149	0.1992	0.3899
Hi-UrWD-USLM+UWLMWD -No-Reordering	4.8149	7.1379	0.1994	0.3917
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering	3.1653	6.8250	0.0722	0.3417
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering	3.1653	6.8250	0.0722	0.3417
Hi-UrWOD-UWLMWOD-With-Reordering	3.6353	6.6375	0.0935	0.3244
HiWOD-Ur-UWLMWOD-No-Reordering	3.6353	6.6375	0.0935	0.3244
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering	3.4950	6.9415	0.0834	0.3367
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering	3.4950	6.9415	0.0834	0.3367
Hi-UrWOD-USLM-With-Reordering	3.5886	6.8023	0.0928	0.3291
Hi-UrWOD-USLM-No-Reordering	3.6255	6.9379	0.0933	0.3405
Hi-UrWOD-USLM-Tuned-With-Reordering	3.5376	6.9892	0.0847	0.3430
Hi-UrWOD-USLM-Tuned-No-Reordering	3.5376	7.0137	0.0847	0.3449
Hi-UrWOD-USLM+UWLMWOD-With-Reordering	3.6794	7.4576	0.0950	0.4017
Hi-UrWOD-USLM+UWLMWOD -No-Reordering	3.6894	7.4737	0.0952	0.4031
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering	3.5449	6.9911	0.0847	0.3417
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering	3.5449	6.9949	0.0847	0.3418

Table 91: HU Test Set 2 results of Hindi toUrdu SMT System (Cluster Alignments)

Models	NIST Score		BLEU Score	
	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering	2.2439	6.9533	0.0214	0.3419
Hi-UrWD-UWLMWD-No-Reordering	2.2415	6.9627	0.0214	0.3425
Hi-UrWD-UWLMWD-Tuned-With-Reordering	3.4667	6.5750	0.0887	0.2988
Hi-UrWD-UWLMWD-Tuned-No-Reordering	3.4667	6.5750	0.0887	0.2988
Hi-UrWD-USLM-With-Reordering	5.3530	7.2819	0.1969	0.3626
Hi-UrWD-USLM-No-Reordering	5.4020	7.3960	0.2001	0.3730
Hi-UrWD-USLM-Tuned-With-Reordering	3.6150	6.7348	0.0970	0.3094
Hi-UrWD-USLM-Tuned-No-Reordering	3.6193	6.7297	0.0974	0.3090
Hi-UrWD-USLM+UWLMWD-With-Reordering	5.7124	7.6774	0.2362	0.4237
Hi-UrWD-USLM+UWLMWD -No-Reordering	5.7212	7.7119	0.2367	0.4270
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering	3.4138	6.5460	0.0857	0.2959
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering	3.4138	6.5424	0.0857	0.2959
Hi-UrWOD-UWLMWOD-With-Reordering	3.7810	6.7270	0.0958	0.3241
HiWOD-Ur-UWLMWOD-No-Reordering	3.7794	6.7219	0.0957	0.3240
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering	3.5116	6.7420	0.0781	0.3084
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering	3.5116	6.7420	0.0781	0.3084
Hi-UrWOD-USLM-With-Reordering	3.6878	7.3158	0.0797	0.3621
Hi-UrWOD-USLM-No-Reordering	3.6839	7.3208	0.0796	0.3624
Hi-UrWOD-USLM-Tuned-With-Reordering	3.6711	7.0923	0.0835	0.3370
Hi-UrWOD-USLM-Tuned-No-Reordering	3.6696	7.0918	0.0834	0.3371
Hi-UrWOD-USLM+UWLMWOD-With-Reordering	3.7594	7.5148	0.0875	0.3859
Hi-UrWOD-USLM+UWLMWOD -No-Reordering	3.7570	7.5197	0.0875	0.3860
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering	3.5871	6.9340	0.0798	0.3250
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering	3.5864	6.9351	0.0797	0.3250

Table 92: HU Test Set 3 results of Hindi toUrdu SMT System (Character Alignments)

Models	NIST Score		BLEU Score	
	default output	processed output	default output	processed output
Hi-UrWD-UWLMWD-With-Reordering	3.1516	6.3242	0.0798	0.2933
Hi-UrWD-UWLMWD-No-Reordering	3.1516	6.3215	0.0798	0.2930
Hi-UrWD-UWLMWD-Tuned-With-Reordering	3.3308	6.3589	0.0903	0.2981
Hi-UrWD-UWLMWD-Tuned-No-Reordering	3.3308	6.3541	0.0903	0.2972
Hi-UrWD-USLM-With-Reordering	5.5061	6.9699	0.2260	0.3601
Hi-UrWD-USLM-No-Reordering	5.5228	7.0220	0.2273	0.3648
Hi-UrWD-USLM-Tuned-With-Reordering	3.1461	6.1437	0.0762	0.2658
Hi-UrWD-USLM-Tuned-No-Reordering	3.1740	6.1795	0.0784	0.2710
Hi-UrWD-USLM+UWLMWD-With-Reordering	5.3055	6.7717	0.2115	0.3411
Hi-UrWD-USLM+UWLMWD -No-Reordering	5.3158	6.7960	0.2120	0.3434
Hi-UrWD-USLM+UWLMWD -Tuned-With-Reordering	3.3181	6.4043	0.0895	0.3008
Hi-UrWD-USLM+UWLMWD -Tuned-No-Reordering	3.3181	6.4016	0.0895	0.3006
Hi-UrWOD-UWLMWOD-With-Reordering	3.7729	6.7817	0.0949	0.3286
HiWOD-Ur-UWLMWOD-No-Reordering	3.7729	6.7817	0.0949	0.3286
Hi-UrWOD-UWLMWOD-Tuned-With-Reordering	3.5158	6.7927	0.0788	0.3145
Hi-UrWOD-UWLMWOD-Tuned-No-Reordering	3.5158	6.7953	0.0788	0.3147
Hi-UrWOD-USLM-With-Reordering	3.7471	7.3287	0.0824	0.3662
Hi-UrWOD-USLM-No-Reordering	3.7423	7.3443	0.0822	0.3664
Hi-UrWOD-USLM-Tuned-With-Reordering	3.5646	6.8894	0.0809	0.3234
Hi-UrWOD-USLM-Tuned-No-Reordering	3.5646	6.8946	0.0809	0.3236
Hi-UrWOD-USLM+UWLMWOD-With-Reordering	3.7952	7.4885	0.0904	0.3854
Hi-UrWOD-USLM+UWLMWOD -No-Reordering	3.7920	7.4924	0.0903	0.3854
Hi-UrWOD-USLM+UWLMWOD -Tuned-With-Reordering	3.5798	6.9320	0.0811	0.3279
Hi-UrWOD-USLM+UWLMWOD -Tuned-No-Reordering	3.5798	6.9346	0.0811	0.3280

Table 93: HU Test Set 3 results of Hindi toUrdu SMT System (Cluster Alignments)

Urdu to Hindi Results

Character alignment results

Models	Sentence Accuracy		Word accuracy		Character accuracy	
	With diacritics	Without diacritics	With diacritics	Without diacritics	With diacritics	Without diacritics
UrWD-Hi-HWLM-With-Reordering			65.0%	32.4%	93%	80.9%
UrWD-Hi-HWLM-No-Reordering			65.2%	33.2%	93.1%	81.6%
UrWD-Hi-HWLM-Tuned-With-Reordering			79.9%	35.2%	96%	81%
UrWD-Hi-HWLM-Tuned-No-Reordering			79.9%	35.2%	96%	81%
UrWD-Hi-HSLM-With-Reordering			48.5%	26.2%	88.5%	75.9%
UrWD-Hi-HSLM-No-Reordering			49.9%	27.4%	89.4%	78.4%
UrWD-Hi-HSLM-Tuned-With-Reordering			78.3%	32.8%	95.6%	79.3%
UrWD-Hi-HSLM-Tuned-No-Reordering			78.3%	32.8%	95.6%	79.3%
UrWD-Hi-HSLM+HWLM-With-Reordering			66.1%	34.3%	93.2%	80.7%
UrWD-Hi-HSLM+HWLM -No-Reordering			66.8%	35.0%	93.5%	81.7%
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering			80.0%	37.5%	96.2%	81%
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering			80.0%	37.5%	96.2%	81%
UrWOD-Hi-HWLM-With-Reordering			7.5%	28.9%	79.9%	85.8%
UrWOD-Hi-HWLM-No-Reordering			7.5%	29.0%	82.3%	86%
UrWOD-Hi-HWLM-Tuned-With-Reordering			9.6%	44.0%	82.6%	85.6%
UrWOD-Hi-HWLM-Tuned-No-Reordering			9.6%	44.0%	82.6%	85.6%
UrWOD-Hi-HSLM-With-Reordering			6.4%	25.8%	76.2%	80.3%
UrWOD-Hi-HSLM-No-Reordering			6.4%	26.2%	78.9%	81.2%
UrWOD-Hi-HSLM-Tuned-With-Reordering			9.2%	43.1%	81.6%	84.6%
UrWOD-Hi-HSLM-Tuned-No-Reordering			9.2%	43.1%	81.6%	84.6%
UrWOD-Hi-HSLM+HWLM-With-Reordering			8.3%	35.4%	79.9%	84.8%
UrWOD-Hi-HSLM+HWLM -No-Reordering			8.3%	35.6%	81.8%	85%
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering			9.7%	45.7%	82.1%	85.6%
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering			9.7%	45.7%	82.2%	85.6%

Table 94: HU Test Set 1 results of Urdu to Hindi SMT systems (character alignments)

Models	Sentence Accuracy		Word accuracy		Character accuracy	
	With diacritics	**Without diacritics**	**With diacritics**	**Without diacritics**	**With diacritics**	**Without diacritics**
UrWD-Hi-HWLM-With-Reordering	2.5%	1.5%	57.4%	52.6%	90.8%	86.7%
UrWD-Hi-HWLM-No-Reordering	2.5%	1.5%	57.5%	52.7%	90.2%	86.8%
UrWD-Hi-HWLM-Tuned-With-Reordering	4%	2%	65.9%	52.1%	91.3%	85.1%
UrWD-Hi-HWLM-Tuned-No-Reordering	4%	2%	65.9%	52.1%	91.3%	85.1%
UrWD-Hi-HSLM-With-Reordering	3%	2%	67.9%	61.8%	93.5%	88.4%
UrWD-Hi-HSLM-No-Reordering	3%	2%	68.0%	62.1%	92%	88.8%
UrWD-Hi-HSLM-Tuned-With-Reordering	4.5%	2%	68.2%	53.9%	91.2%	84.7%
UrWD-Hi-HSLM-Tuned-No-Reordering	4.5%	2%	68.2%	53.9%	91.2%	84.7%
UrWD-Hi-HSLM+HWLM-With-Reordering	3%	1.5%	65.5%	59.7%	92.4%	88.2%
UrWD-Hi-HSLM+HWLM -No-Reordering	3%	1.5%	65.5%	59.8%	91%	88.4%
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering	5.5%	2%	72.2%	57.9%	91.8%	85.8%
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering	5.5%	2%	72.2%	57.9%	91.8%	85.8%
UrWOD-Hi-HWLM-With-Reordering	0.5%	1%	37.2%	52.3%	84.2%	89.2%
UrWOD-Hi-HWLM-No-Reordering	0.5%	1%	37.2%	52.3%	83.8%	89.2%
UrWOD-Hi-HWLM-Tuned-With-Reordering	0.5%	1.5%	44.0%	58.7%	83.4%	88.4%
UrWOD-Hi-HWLM-Tuned-No-Reordering	0.5%	1.5%	44.0%	58.7%	83.4%	88.4%
UrWOD-Hi-HSLM-With-Reordering	0.5%	4%	48.8%	75.3%	86.8%	94.3%
UrWOD-Hi-HSLM-No-Reordering	0.5%	4%	48.9%	75.4%	84.5%	94.3%
UrWOD-Hi-HSLM-Tuned-With-Reordering	0.5%	2.5%	45.6%	66.1%	82.8%	90.3%
UrWOD-Hi-HSLM-Tuned-No-Reordering	0.5%	2.5%	45.6%	66.1%	82.8%	90.3%
UrWOD-Hi-HSLM+HWLM-With-Reordering	0.5%	5%	50.1%	77.0%	86.8%	94.6%
UrWOD-Hi-HSLM+HWLM -No-Reordering	0.5%	5%	50.1%	77.0%	85.3%	94.6%
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering	1%	3%	49.9%	72.1%	84.3%	91.9%
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering	1%	3%	49.9%	72.1%	84.3%	91.9%

Table 95: HU Test Set 2 results of Urdu to Hindi SMT systems (character alignments)

Models	Sentence Accuracy		Word accuracy		Character accuracy	
	With diacritics	Without diacritics	With diacritics	Without diacritics	With diacritics	Without diacritics
UrWD-Hi-HWLM-With-Reordering	0.9%	0.4%	59.2%	45.7%	90.9%	85.8%
UrWD-Hi-HWLM-No-Reordering	0.9%	0.4%	59.3%	45.7%	91.1%	85.9%
UrWD-Hi-HWLM-Tuned-With-Reordering	4%	0.4%	74.0%	50.4%	94%	86.3%
UrWD-Hi-HWLM-Tuned-No-Reordering	4%	0.4%	74.0%	50.4%	94%	86.3%
UrWD-Hi-HSLM-With-Reordering	2.2%	0	64.2%	50.0%	91.3%	85.4%
UrWD-Hi-HSLM-No-Reordering	2.2%	0	64.7%	50.3%	92.1%	86.1%
UrWD-Hi-HSLM-Tuned-With-Reordering	2.7%	0	73.8%	49.7%	93.7%	85.6%
UrWD-Hi-HSLM-Tuned-No-Reordering	2.7%	0	73.8%	49.7%	93.7%	85.6%
UrWD-Hi-HSLM+HWLM-With-Reordering	1.3%	0	65.2%	48.7%	92.4%	86.4%
UrWD-Hi-HSLM+HWLM -No-Reordering	1.3%	0	65.2%	48.8%	92.4%	86.6%
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering	5.3%	0.4%	77.8%	53.8%	94.4%	86.7%
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering	5.3%	0.4%	77.8%	53.8%	94.4%	86.7%
UrWOD-Hi-HWLM-With-Reordering	0	0	33.2%	42.5%	86.1%	87.8%
UrWOD-Hi-HWLM-No-Reordering	0	0	33.2%	42.5%	86.2%	87.8%
UrWOD-Hi-HWLM-Tuned-With-Reordering	0	0	42.1%	51.8%	86.3%	88.1%
UrWOD-Hi-HWLM-Tuned-No-Reordering	0	0	42.1%	51.8%	86.3%	88.1%
UrWOD-Hi-HSLM-With-Reordering	0	0.9%	42.5%	56.2%	85%	89.2%
UrWOD-Hi-HSLM-No-Reordering	0	0.9%	42.5%	56.3%	85.5%	89.3%
UrWOD-Hi-HSLM-Tuned-With-Reordering	0	0.4%	43.9%	56.0%	85.8%	88.8%
UrWOD-Hi-HSLM-Tuned-No-Reordering	0	0.4%	43.9%	56.0%	85.8%	88.8%
UrWOD-Hi-HSLM+HWLM-With-Reordering	0	0.4%	44.8%	60.1%	87.3%	90.8%
UrWOD-Hi-HSLM+HWLM -No-Reordering	0	0.4%	44.8%	60.1%	87.7%	90.8%
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering	0	0	47.2%	59.9%	86.8%	89.9%
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering	0	0	47.2%	59.9%	86.9%	89.9%

Table 96: HU Test Set 3 results of Urdu to Hindi SMT systems (character alignments)

Cluster alignment results

Models	Sentence Accuracy		Word accuracy		Character accuracy	
	With diacritics	Without diacritics	With diacritics	Without diacritics	With diacritics	Without diacritics
UrWD-Hi-HWLM-With-Reordering	72.5%	33.6%	72.5%	33.6%	94.4%	81.9%
UrWD-Hi-HWLM-No-Reordering	72.9%	34.3%	72.9%	34.3%	94.5%	82.6%
UrWD-Hi-HWLM-Tuned-With-Reordering	81.2%	34.8%	81.2%	34.8%	96.1%	81.5%
UrWD-Hi-HWLM-Tuned-No-Reordering	81.2%	34.9%	81.2%	34.9%	96.1%	81.6%
UrWD-Hi-HSLM-With-Reordering	55.2%	26.9%	55.2%	26.9%	90.4%	77.2%
UrWD-Hi-HSLM-No-Reordering	56.9%	28.3%	56.9%	28.3%	91.2%	79.2%
UrWD-Hi-HSLM-Tuned-With-Reordering	79.3%	34.1%	79.3%	34.1%	95.6%	80.2%
UrWD-Hi-HSLM-Tuned-No-Reordering	789.3%	43.1%	79.3%	34.1%	95.6%	80.2%
UrWD-Hi-HSLM+HWLM-With-Reordering	71.3%	34.1%	71.3%	34.1%	94.3%	81.4%
UrWD-Hi-HSLM+HWLM -No-Reordering	71.8%	34.7%	71.8%	34.7%	94.6%	82.3%
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering	82.7%	36.1%	82.8%	36.1%	96.4%	81.4%
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering	82.8%	36.1%	82.8%	36.1%	96.4%	81.5%
UrWOD-Hi-HWLM-With-Reordering	8.3%	32.5%	8.3%	32.5%	78%	86.3%
UrWOD-Hi-HWLM-No-Reordering	8.3%	32.6%	8.3%	32.6%	79.9%	86.5%
UrWOD-Hi-HWLM-Tuned-With-Reordering	9.2%	43.1%	9.2%	43.1%	79.4%	85.6%
UrWOD-Hi-HWLM-Tuned-No-Reordering	9.2%	43.1%	9.2%	43.1%	79.4%	85.6%
UrWOD-Hi-HSLM-With-Reordering	7%	28.1%	7.0%	28.1%	74.3%	80.4%
UrWOD-Hi-HSLM-No-Reordering	7.1%	28.6%	7.1%	28.6%	77.4%	81.5%
UrWOD-Hi-HSLM-Tuned-With-Reordering	9.2%	43.6%	9.2%	43.6%	77.3%	84.6%
UrWOD-Hi-HSLM-Tuned-No-Reordering	9.2%	43.6%	9.2%	43.6%	79.3%	84.6%
UrWOD-Hi-HSLM+HWLM-With-Reordering	8.4%	37.1%	8.4%	37.1%	77.3%	84.8%
UrWOD-Hi-HSLM+HWLM -No-Reordering	8.5%	37.4%	8.5%	37.4%	79.7%	85.2%
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering	9.6%	46.2%	9.6%	46.2%	76.7%	86.2%
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering	9.5%	46.1%	9.5%	46.1%	79.8%	86.2%

Table 97: HU Test Set 1 results of Urdu to Hindi SMT systems (cluster alignments)

Models	Sentence Accuracy		Word accuracy		Character accuracy	
	With diacritics	Without diacritics	With diacritics	Without diacritics	With diacritics	Without diacritics
UrWD-Hi-HWLM-With-Reordering	3.5%	1.5%	61.6%	51.8%	91.7%	86.7%
UrWD-Hi-HWLM-No-Reordering	3.5%	1.5%	61.7%	51.8%	91.1%	86.8%
UrWD-Hi-HWLM-Tuned-With-Reordering	4%	2%	64.4%	52.0%	91%	85.3%
UrWD-Hi-HWLM-Tuned-No-Reordering	4%	2%	64.4%	52.0%	91%	85.3%
UrWD-Hi-HSLM-With-Reordering	5%	2%	69.5%	60.7%	94.1%	88.1%
UrWD-Hi-HSLM-No-Reordering	5%	2%	69.6%	60.8%	92.3%	88.5%
UrWD-Hi-HSLM-Tuned-With-Reordering	4.5%	2%	68.1%	55.0%	91.3%	85.3%
UrWD-Hi-HSLM-Tuned-No-Reordering	4.5%	2%	68.1%	55.0%	91.3%	85.3%
UrWD-Hi-HSLM+HWLM-With-Reordering	4.5%	1.5%	68.6%	59.2%	93.4%	88.3%
UrWD-Hi-HSLM+HWLM -No-Reordering	4.5%	1.5%	68.6%	59.2%	92.1%	88.4%
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering	5%	2%	69.5%	57.4%	91.6%	85.9%
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering	5%	2%	69.5%	57.4%	91.6%	85.9%
UrWOD-Hi-HWLM-With-Reordering	0.5%	1%	38.4%	55.0%	81.6%	89.7%
UrWOD-Hi-HWLM-No-Reordering	0.5%	1%	38.4%	55.0%	81.6%	89.7%
UrWOD-Hi-HWLM-Tuned-With-Reordering	0.5%	1%	43.0%	57.9%	80.7%	88%
UrWOD-Hi-HWLM-Tuned-No-Reordering	0.5%	1%	43.0%	57.9%	80.7%	88%
UrWOD-Hi-HSLM-With-Reordering	0.5%	4%	49.2%	75.9%	84.3%	94.3%
UrWOD-Hi-HSLM-No-Reordering	0.5%	4%	49.2%	76.0%	82.6%	94.3%
UrWOD-Hi-HSLM-Tuned-With-Reordering	1%	3%	50.4%	70.2%	81.5%	91.1%
UrWOD-Hi-HSLM-Tuned-No-Reordering	1%	3%	50.4%	70.2%	81.2%	91.1%
UrWOD-Hi-HSLM+HWLM-With-Reordering	0.5%	5%	50.7%	77.9%	84.1%	94.7%
UrWOD-Hi-HSLM+HWLM -No-Reordering	0.5%	5%	50.7%	77.9%	83.1%	94.7%
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering	0.5%	2%	47.1%	69.1%	81.1%	91.3%
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering	0.5%	2%	47.1%	69.2%	82%	91.3%

Table 98: HU Test Set 2 results of Urdu to Hindi SMT systems (cluster alignments)

Models	Sentence Accuracy		Word accuracy		Character accuracy	
	With diacritics	Without diacritics	With diacritics	Without diacritics	With diacritics	Without diacritics
UrWD-Hi-HWLM-With-Reordering	1.8%	0.4%	66.2%	46.1%	92.5%	86%
UrWD-Hi-HWLM-No-Reordering	1.8%	0.4%	66.3%	46.2%	92.5%	86%
UrWD-Hi-HWLM-Tuned-With-Reordering	2.2%	0.4%	71.7%	50.4%	93.5%	86.6%
UrWD-Hi-HWLM-Tuned-No-Reordering	2.2%	0	71.7%	50.4%	93.5%	86.6%
UrWD-Hi-HSLM-With-Reordering	1.8%	0	66.0%	49.0%	91.9%	85.1%
UrWD-Hi-HSLM-No-Reordering	1.8%	0	66.5%	49.4%	92.4%	85.8%
UrWD-Hi-HSLM-Tuned-With-Reordering	2.7%	0	71.7%	49.8%	93.3%	86%
UrWD-Hi-HSLM-Tuned-No-Reordering	2.7%	0	71.7%	49.8%	93.3%	86%
UrWD-Hi-HSLM+HWLM-With-Reordering	2.7%	0	69.0%	48.8%	93.3%	86.4%
UrWD-Hi-HSLM+HWLM -No-Reordering	2.7%	0	69.1%	48.8%	93.1%	86.4%
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering	3.5%	0.4%	75.1%	53.4%	93.8%	86.7%
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering	3.5%	0.4%	75.1%	53.4%	93.8%	86.7%
UrWOD-Hi-HWLM-With-Reordering	0	0.4%	34.6%	47.2%	82.7%	88.4%
UrWOD-Hi-HWLM-No-Reordering	0	0.4%	34.6%	47.2%	83.1%	88.4%
UrWOD-Hi-HWLM-Tuned-With-Reordering	0	0	41.2%	52.7%	83%	88%
UrWOD-Hi-HWLM-Tuned-No-Reordering	0	0	41.2%	52.7%	83%	88%
UrWOD-Hi-HSLM-With-Reordering	0	0.4%	42.9%	56.4%	82.9%	89.7%
UrWOD-Hi-HSLM-No-Reordering	0	0.4%	42.9%	56.5%	83.7%	89%
UrWOD-Hi-HSLM-Tuned-With-Reordering	0	0.4%	48.1%	59.5%	84.7%	88.4%
UrWOD-Hi-HSLM-Tuned-No-Reordering	0	0.4%	48.1%	59.5%	85%	89.7%
UrWOD-Hi-HSLM+HWLM-With-Reordering	0	0.4%	45.9%	60.9%	84.7%	90.5%
UrWOD-Hi-HSLM+HWLM -No-Reordering	0	0.4%	45.9%	60.9%	85.2%	90.5%
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering	0	0	44.5%	57.2%	83.5%	89.3%
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering	0	0	44.5%	57.2%	84.5%	89.3%

Table 99: HU Test Set 3 results of Urdu to Hindi SMT systems (cluster alignments)

NIST and BLEU Scores

In this section, we will give n-gram measures for our Urdu to Hindi SMT systems.

Models	NIST Scores		BLEU Scores	
	With diacritics	Without diacritics	With diacritics	Without diacritics
UrWD-Hi-HWLM-With-Reordering	7.3106	3.6209	0	0
UrWD-Hi-HWLM-No-Reordering	7.3396	3.7122	0	0
UrWD-Hi-HWLM-Tuned-With-Reordering	8.9915	3.9369	0	0
UrWD-Hi-HWLM-Tuned-No-Reordering	8.9915	3.9369	0	0
UrWD-Hi-HSLM-With-Reordering	5.4505	2.9171	0	0
UrWD-Hi-HSLM-No-Reordering	5.6023	3.0575	0	0
UrWD-Hi-HSLM-Tuned-With-Reordering	8.8185	3.6605	0	0
UrWD-Hi-HSLM-Tuned-No-Reordering	8.8185	3.6605	0	0
UrWD-Hi-HSLM+HWLM-With-Reordering	7.4363	3.8341	0	0
UrWD-Hi-HSLM+HWLM -No-Reordering	7.5110	3.9172	0	0
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering	9.2328	4.1958	0	0
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering	9.2328	4.1958	0	0
UrWOD-Hi-HWLM-With-Reordering	0.8191	3.2329	0	0
UrWOD-Hi-HWLM-No-Reordering	0.8191	3.2371	0	0
UrWOD-Hi-HWLM-Tuned-With-Reordering	1.0532	4.9269	0	0
UrWOD-Hi-HWLM-Tuned-No-Reordering	1.0532	4.9269	0	0
UrWOD-Hi-HSLM-With-Reordering	0.6963	2.8871	0	0
UrWOD-Hi-HSLM-No-Reordering	0.6963	2.9238	0	0
UrWOD-Hi-HSLM-Tuned-With-Reordering	0.0113	4.8232	0	0
UrWOD-Hi-HSLM-Tuned-No-Reordering	0.0113	4.8232	0	0
UrWOD-Hi-HSLM+HWLM-With-Reordering	0.9079	3.9601	0	0
UrWOD-Hi-HSLM+HWLM -No-Reordering	0.9121	3.9809	0	0
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering	1.0684	5.1185	0	0
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering	1.0684	5.1185	0	0

Table 100: HU Test Set 1 results of Urdu to Hindi SMT systems (character alignments)

Models	NIST Scores		BLEU Scores	
	With diacritics	Without diacritics	With diacritics	Without diacritics
UrWD-Hi-HWLM-With-Reordering	8.1570	3.7564	0	0
UrWD-Hi-HWLM-No-Reordering	8.1985	3.8353	0	0
UrWD-Hi-HWLM-Tuned-With-Reordering	9.1377	3.8946	0	0
UrWD-Hi-HWLM-Tuned-No-Reordering	9.1377	3.9029	0	0
UrWD-Hi-HSLM-With-Reordering	6.2001	2.9970	0	0
UrWD-Hi-HSLM-No-Reordering	6.3942	3.1567	0	0
UrWD-Hi-HSLM-Tuned-With-Reordering	8.9306	3.8092	0	0
UrWD-Hi-HSLM-Tuned-No-Reordering	8.9306	3.8092	0	0
UrWD-Hi-HSLM+HWLM-With-Reordering	8.0164	3.8156	0	0
UrWD-Hi-HSLM+HWLM -No-Reordering	8.0787	3.8778	0	0
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering	9.3117	4.0368	0	0
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering	9.3200	4.0368	0	0
UrWOD-Hi-HWLM-With-Reordering	0.8915	3.6336	0	0
UrWOD-Hi-HWLM-No-Reordering	0.8955	3.6419	0	0
UrWOD-Hi-HWLM-Tuned-With-Reordering	0.9970	4.8280	0	0
UrWOD-Hi-HWLM-Tuned-No-Reordering	0.9970	4.8280	0	0
UrWOD-Hi-HSLM-With-Reordering	0.7503	3.1419	0	0
UrWOD-Hi-HSLM-No-Reordering	0.7618	3.1993	0	0
UrWOD-Hi-HSLM-Tuned-With-Reordering	0.9868	4.8826	0	0
UrWOD-Hi-HSLM-Tuned-No-Reordering	0.9868	4.8826	0	0
UrWOD-Hi-HSLM+HWLM-With-Reordering	0.9030	4.1460	0	0
UrWOD-Hi-HSLM+HWLM -No-Reordering	0.9111	4.1834	0	0
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering	1.0337	5.1789	0	0
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering	1.0297	5.1748	0	0

Table 101: HU Test Set 1 results of Urdu to Hindi SMT systems (cluster alignments)

Models	NIST Scores		BLEU Scores	
	With diacritics	Without diacritics	With diacritics	Without diacritics
UrWD-Hi-HWLM-With-Reordering	5.7115	5.0324	0.2391	0.1820
UrWD-Hi-HWLM-No-Reordering	5.7246	5.0441	0.2401	0.1823
UrWD-Hi-HWLM-Tuned-With-Reordering	6.6006	4.8655	0.3389	0.1750
UrWD-Hi-HWLM-Tuned-No-Reordering	6.6022	4.8655	0.3392	0.1750
UrWD-Hi-HSLM-With-Reordering	6.9156	6.1713	0.3568	0.2785
UrWD-Hi-HSLM-No-Reordering	6.9445	6.2082	0.3603	0.2819
UrWD-Hi-HSLM-Tuned-With-Reordering	6.9042	5.0990	0.3663	0.1947
UrWD-Hi-HSLM-Tuned-No-Reordering	6.9042	5.0990	0.3663	0.1947
UrWD-Hi-HSLM+HWLM-With-Reordering	6.7530	5.9653	0.3275	0.2552
UrWD-Hi-HSLM+HWLM -No-Reordering	6.7603	5.9681	0.3287	0.2558
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering	7.4260	5.5867	0.4363	0.2328
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering	7.4260	5.5867	0.4363	0.2328
UrWOD-Hi-HWLM-With-Reordering	3.0849	5.0815	0.0661	0.1895
UrWOD-Hi-HWLM-No-Reordering	3.0856	5.0815	0.0662	0.1895
UrWOD-Hi-HWLM-Tuned-With-Reordering	3.8962	5.8543	0.1075	0.2396
UrWOD-Hi-HWLM-Tuned-No-Reordering	3.8962	5.8543	0.1075	0.2396
UrWOD-Hi-HSLM-With-Reordering	4.3257	8.1055	0.1353	0.4856
UrWOD-Hi-HSLM-No-Reordering	4.3311	8.1173	0.1354	0.4871
UrWOD-Hi-HSLM-Tuned-With-Reordering	4.1129	6.8617	0.1190	0.3434
UrWOD-Hi-HSLM-Tuned-No-Reordering	4.1129	6.8617	0.1190	0.3434
UrWOD-Hi-HSLM+HWLM-With-Reordering	4.4933	8.3579	0.1495	0.5221
UrWOD-Hi-HSLM+HWLM -No-Reordering	4.4944	8.3579	0.1495	0.5221
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering	4.4721	7.6385	0.1471	0.4347
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering	4.4731	7.6385	0.1471	0.4347

Table 102: HU Test Set 2 results of Urdu to Hindi SMT systems (character alignments)

Models	NIST Scores		BLEU Scores	
	With diacritics	Without diacritics	With diacritics	Without diacritics
UrWD-Hi-HWLM-With-Reordering	6.1140	4.9280	0.2912	0.1753
UrWD-Hi-HWLM-No-Reordering	6.1272	4.9444	0.2925	0.1760
UrWD-Hi-HWLM-Tuned-With-Reordering	6.4231	4.8720	0.3234	0.1718
UrWD-Hi-HWLM-Tuned-No-Reordering	6.4246	4.8720	0.3237	0.1718
UrWD-Hi-HSLM-With-Reordering	7.0765	6.0199	0.3904	0.2662
UrWD-Hi-HSLM-No-Reordering	7.0994	6.0430	0.3942	0.2693
UrWD-Hi-HSLM-Tuned-With-Reordering	6.9060	5.2401	0.3707	0.2015
UrWD-Hi-HSLM-Tuned-No-Reordering	6.9060	5.2401	0.3707	0.2015
UrWD-Hi-HSLM+HWLM-With-Reordering	7.0374	5.8839	0.3820	0.2511
UrWD-Hi-HSLM+HWLM -No-Reordering	7.0463	5.8884	0.3834	0.2512
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering	7.0917	5.5505	0.3970	0.2296
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering	7.0934	5.5505	0.3973	0.2296
UrWOD-Hi-HWLM-With-Reordering	3.2368	5.3894	0.0729	0.2150
UrWOD-Hi-HWLM-No-Reordering	3.2376	5.3894	0.0729	0.2150
UrWOD-Hi-HWLM-Tuned-With-Reordering	3.8123	5.7376	0.1016	0.2319
UrWOD-Hi-HWLM-Tuned-No-Reordering	3.8123	5.7376	0.1016	0.2319
UrWOD-Hi-HSLM-With-Reordering	4.3812	8.2119	0.1392	0.4954
UrWOD-Hi-HSLM-No-Reordering	4.3867	8.2208	0.1394	0.4968
UrWOD-Hi-HSLM-Tuned-With-Reordering	4.5328	7.3296	0.1501	0.4070
UrWOD-Hi-HSLM-Tuned-No-Reordering	4.5338	7.3296	0.1501	0.4070
UrWOD-Hi-HSLM+HWLM-With-Reordering	4.5599	8.4932	0.1523	0.5372
UrWOD-Hi-HSLM+HWLM -No-Reordering	4.5610	8.4932	0.1523	0.5372
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering	4.2725	7.2652	0.1308	0.3867
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering	4.2735	7.2734	0.1309	0.3871

Table 103: HU Test Set 2 results of Urdu to Hindi SMT systems (cluster alignments)

Models	NIST Scores		BLEU Scores	
	With diacritics	Without diacritics	With diacritics	Without diacritics
UrWD-Hi-HWLM-With-Reordering	6.3125	4.5042	0.2826	0.1329
UrWD-Hi-HWLM-No-Reordering	6.1378	4.5095	0.2829	0.1331
UrWD-Hi-HWLM-Tuned-With-Reordering	7.9140	4.9542	0.4581	0.1726
UrWD-Hi-HWLM-Tuned-No-Reordering	7.9140	4.9582	0.4581	0.1726
UrWD-Hi-HSLM-With-Reordering	6.6686	4.8693	0.3301	0.1638
UrWD-Hi-HSLM-No-Reordering	6.7433	4.9092	0.3355	0.1652
UrWD-Hi-HSLM-Tuned-With-Reordering	7.8936	4.8541	0.4542	0.1614
UrWD-Hi-HSLM-Tuned-No-Reordering	7.8936	4.8541	0.4542	0.1614
UrWD-Hi-HSLM+HWLM-With-Reordering	7.0345	4.8709	0.3538	0.1528
UrWD-Hi-HSLM+HWLM -No-Reordering	7.0377	4.8740	0.3541	0.1531
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering	8.4280	5.3472	0.5180	0.1981
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering	8.4280	5.3472	0.5180	0.1981
UrWOD-Hi-HWLM-With-Reordering	2.9901	4.1696	0.0548	0.1147
UrWOD-Hi-HWLM-No-Reordering	2.9901	4.1696	0.0548	0.1147
UrWOD-Hi-HWLM-Tuned-With-Reordering	3.9224	5.2526	0.0964	0.1889
UrWOD-Hi-HWLM-Tuned-No-Reordering	3.9224	5.2526	0.0964	0.1889
UrWOD-Hi-HSLM-With-Reordering	3.8282	5.6215	0.1015	0.2360
UrWOD-Hi-HSLM-No-Reordering	3.8378	5.6338	0.1022	0.2360
UrWOD-Hi-HSLM-Tuned-With-Reordering	4.1884	5.7510	0.1155	0.2310
UrWOD-Hi-HSLM-Tuned-No-Reordering	4.1884	5.7510	0.1155	0.2310
UrWOD-Hi-HSLM+HWLM-With-Reordering	4.1357	6.1735	0.1149	0.2837
UrWOD-Hi-HSLM+HWLM -No-Reordering	4.1357	6.1735	0.1149	0.2837
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering	4.4350	6.1203	0.1288	0.2656
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering	4.4350	6.1203	0.1288	0.2656

Table 104: HU Test Set 3 results of Urdu to Hindi SMT systems (character alignments)

Models	NIST Scores		BLEU Scores	
	With diacritics	Without diacritics	With diacritics	Without diacritics
UrWD-Hi-HWLM-With-Reordering	6.9270	5.5413	0.3391	0.1340
UrWD-Hi-HWLM-No-Reordering	6.9353	4.5505	0.3400	0.1342
UrWD-Hi-HWLM-Tuned-With-Reordering	7.6400	4.9599	0.4280	0.1737
UrWD-Hi-HWLM-Tuned-No-Reordering	7.6400	4.9639	0.4280	0.1738
UrWD-Hi-HSLM-With-Reordering	6.7390	4.7316	0.3349	0.1533
UrWD-Hi-HSLM-No-Reordering	6.8219	4.7887	0.3422	0.1568
UrWD-Hi-HSLM-Tuned-With-Reordering	7.6529	4.8742	0.4273	0.1620
UrWD-Hi-HSLM-Tuned-No-Reordering	7.6529	4.8742	0.4273	0.1620
UrWD-Hi-HSLM+HWLM-With-Reordering	7.3232	4.8761	0.3831	0.1548
UrWD-Hi-HSLM+HWLM -No-Reordering	7.3322	4.8792	0.3845	0.1551
UrWD-Hi-HSLM+HWLM -Tuned-With-Reordering	8.1287	5.3109	0.4814	0.1916
UrWD-Hi-HSLM+HWLM -Tuned-No-Reordering	8.1287	5.3109	0.4814	0.1916
UrWOD-Hi-HWLM-With-Reordering	3.1570	4.6754	0.0621	0.1505
UrWOD-Hi-HWLM-No-Reordering	3.1570	4.6754	0.0621	0.1505
UrWOD-Hi-HWLM-Tuned-With-Reordering	3.8982	5.3438	0.0951	0.1982
UrWOD-Hi-HWLM-Tuned-No-Reordering	3.8982	5.3438	0.0951	0.1982
UrWOD-Hi-HSLM-With-Reordering	3.8832	5.6539	0.1038	0.2362
UrWOD-Hi-HSLM-No-Reordering	3.8973	5.6706	0.1054	0.2378
UrWOD-Hi-HSLM-Tuned-With-Reordering	4.5634	6.0536	0.1387	0.2598
UrWOD-Hi-HSLM-Tuned-No-Reordering	4.5634	6.0536	0.1387	0.2598
UrWOD-Hi-HSLM+HWLM-With-Reordering	4.2820	6.2695	0.1253	0.2885
UrWOD-Hi-HSLM+HWLM -No-Reordering	4.2820	6.2695	0.1253	0.2885
UrWOD-Hi-HSLM+HWLM -Tuned-With-Reordering	4.2419	5.8788	0.1184	0.2394
UrWOD-Hi-HSLM+HWLM -Tuned-No-Reordering	4.2419	5.8788	0.1184	0.2394

Table 105: HU Test Set 3 results of Urdu to Hindi SMT systems (cluster alignments)

French Summary

For preparing an exended Franch summary of my PhD thesis, we have used AXiMAG system, an inhouse collaborative translation system developed by our GETALP lab. First, we translated the introduction and conclusion of my PhD thesis using AXiMAG system and then post-edited the French translation. Here I will give the English – French parallel text.

Introduction	**Introduction**
In general, the term *translation* is understood as the process of understanding the meaning of a text in one language and subsequently producing an equivalent text in another language, conveying the same message.	En général, le terme *traduction* est compris comme le processus de compréhension du sens d'un texte dans une langue et ensuite de production d'un texte équivalent dans une autre langue, transmettant le même message.
Machine Translation (MT) is a *rêve* of the 1950s [21, 24-26, 80, 125, 171, 172, 178, 179].	La traduction automatique (TA) est un *rêve* des années 1950 [21, 24-26, 80, 125, 171, 172, 178, 179].
Although a large number of milestones have been achieved to enliven the *rêve* of MT [21, 23-26, 28, 32, 41, 80, 87-90, 93, 100-102, 104, 105, 113, 115, 125, 131, 132, 134, 135, 139, 144, 162, 171, 172, 178, 179, 188], it is still a dream in the interdisciplinary research of *computer science* , *artificial intelligence* , *machine learning* , *computational linguistics* , *natural language processing* and *engineering* .	Bien qu'un grand nombre d'étapes importantes aient été réalisées pour animer le *rêve* de la TA [21, 23-26, 28, 32, 41, 80, 87-90, 93, 100-102, 104, 105, 113, 115, 125, 131, 132, 134, 135, 139, 144, 162, 171, 172, 178, 179, 188], c'est toujours un rêve dans la recherche interdisciplinaire en *informatique, intelligence artificielle, apprentissage automatique, linguistique computationnelle, traitement des langues naturelles, et ingénierie.*
The dream of MT is fuzzy like other dreams.	Le rêve de la TA est brouillé, comme d'autres rêves.
To make it crisp, we have to consider precise translation tasks.	Pour le rendre précis, nous devons considérer des tâches précises de traduction.
General purpose, high quality and fully automatic MT is believed to be impossible [10, 24-26].	La TA généraliste, de haute qualité et entièrement automatique est censée être impossible [10, 24-26].
But the general MT problem can be reduced to various subproblems obviously less complex and less hard than the general one.	Mais le problème général de la TA peut être réduit à divers sous-problèmes évidemment moins complexes et moins durs que le problème général.
We will concentrate on a few of them of particular interest, such as *intralingual* or *interdialectal* translation.	Nous nous concentrerons sur quelques-uns d'entre eux, d'un intérêt particulier, tels que la traduction *intralingue* ou *interdialectale.*
That problem reduction can be made on the basis of the domain of application, the sublanguage (a restricted and limited part of a language) considered for translation, the intended users, the language pairs under consideration,	Cette réduction du problème peut être faite sur la base du domaine de l'application, du sous-langage (une partie restreinte et limitée d'une langue) considéré pour la traduction, des utilisateurs prévus, des couples de langues consi-

etc.

These features also help to define the goal and objectives for the subproblems of MT [21, 23-27, 41, 80, 93, 102, 115, 125, 139, 171, 172, 178, 179, 187, 190].

However, these subproblems may still be very hard and complex, although some instances may be quite simple or not so difficult.

We will define later more precisely what we mean by complexity and difficulty.

In any case, there is a long way ahead to go [25, 26, 139, 187].

One of our goals will be to characterize the complexity and difficulty of solving a certain class of translation problems that we will call "*weak translation problems*".

MT is known for its complex nature and *multivalence* is one of its main reasons. *Multivalence* , the term used by Mel'čuk [125], arises due to the non-determinism (*polysemy* during analysis and *synonymy* in generation, and both in transfer) [6, 13, 21, 23-26, 32, 41, 51, 93, 102, 113, 115, 125, 139, 149, 154, 178, 179, 187].

The number of possible translations of an average source language sentence may go up to thousands or in general increase dramatically with its length [6, 13, 21, 25, 26, 41, 93, 113, 125, 139, 149, 154, 178, 179, 187].

Given a source language *SL* and a target language *TL*, a translation unit *S* of *n* words may have an exponential number $T_1, T_2, \dots, T_N$ N = k^n of valid translations $T_1, T_2, \dots, T_N$ in *TL*, where $N = O(k^n)$ for some $k > 1$ depending on the precise subproblem at hand.

To solve the problem of *multivalence* for a given subproblem of MT, different filters are applied at various levels during the phases of preprocessing, analysis, synthesis and post-processing to restrict the cardinality of the possible solution set of the problem to an acceptable and reasonable range of values [6, 13, 21, 23-28, 41, 51, 80, 93, 102, 104, 113, 115, 125, 139, 149, 154, 171, 172, 178, 187].

dérés, *etc.*

Ces fonctionnalités aident également à définir le but et les objectifs des sous-problèmes de la TA [21, 23-27, 41, 80, 93, 102, 115, 125, 139, 171, 172, 178, 179, 187, 190].

Toutefois, ces sous-problèmes peuvent encore être très difficiles et complexes, bien que certaines instances puissent être très simples ou pas aussi difficiles.

Nous définirons plus tard avec plus de précision ce que nous voulons dire par complexité et difficulté.

Dans tous les cas, il ya un long chemin à parcourir pour progresser [25, 26, 139, 187].

Un de nos objectifs sera de caractériser la complexité et la difficulté de résoudre une certaine classe de problèmes de traduction que nous appellerons *« problèmes faibles de traduction ».*

La TA est connue pour son caractère complexe et la *multivalence* (ambiguïté et polysémie) en est l'une des principales raisons. *Multivalence,* le terme utilisé par Mel'čuk [125], apparaît en raison du non-déterminisme *(polysémie* en cours d'analyse et *synonymie* en génération, et les deux en transfert) [6, 13, 21, 23-26, 32, 41, 51, 93, 102, 113, 115, 125, 139, 149, 154, 178, 179, 187].

Le nombre de traductions possibles d'une phrase moyenne en langue source peut aller jusqu'à des milliers ou en général augmenter de façon spectaculaire avec sa longueur [6, 13, 21, 25, 26, 41, 93, 113, 125, 139, 149, 154, 178, 179, 187].

Une langue source *SL* étant donnée, ainsi qu'une langue cible *TL*, une unité de traduction *S* de *n* mots peut avoir un nombre exponentiel $T_1, T_2, \dots, T_N$ de traductions valides, où $N = O(k^n)$ pour un certain $k > 1$ dépendant du sous-problème précis à traiter.

Pour résoudre le problème de la *multivalence* pour un sous-problème donné de TA, différents filtres sont appliqués à différents niveaux pendant les phases de pré-traitement, d'analyse, de synthèse et de post-traitement pour restreindre la cardinalité de l'ensemble des solutions possibles du problème à un plage de valeurs acceptable et raisonnable [6, 13, 21, 23-28, 41, 51, 80, 93, 102, 104, 113, 115, 125, 139, 149, 154, 171, 172, 178, 187].

Transliteration is also a subproblem of MT. It consists in overcoming the scriptural differences among different writing systems used for different languages [1, 3, 4, 9, 15, 16, 47, 50, 57, 59-61, 65, 73, 82-85, 97, 100, 101, 108, 112, 124, 130, 143-146, 150, 153, 165, 168, 174, 181, 189, 191] or even for the same language [118-121, 164].

We are interested in the special class of subproblems of MT π where N is either very small, say always less than 5, or even almost always equal to 1 because of the proximity of the written forms of *SL* and *TL*.

For example, this happens in situations (1) when the languages of a translation pair are extremely close to each other, *e.g.* Bengali–Assamese, Hindi–Marathi, Hindi–Urdu, *etc.*, (2) when translation is performed between two different varieties or dialects of a language, either written in the same writing system (Quebecois–French, Malay–Indonesian) or in mutually unintelligible writing systems (Punjabi, Sindhi, Seraiki) and (3) when the same language is written in different mutually incomprehensible scripts (Kashmiri, Malay, Punjabi, Sindhi, Seraiki).

The domain of our investigation is the class of subproblems π of MT, applied to a pair $\langle(L_i, W_j), (L_k W_l)\rangle$ of combinations of a language and a writing system, such that there exists only one (in most of the cases) or a very small set of valid "translation solutions" to a subproblem π, for a given sentence S of L_i written in W_j.

A natural assumption is that such problems should be very simple (in terms of complexity of the sufficient computational model) and not very difficult (in terms of the human and computation costs involved in preparing the system to perform translation) than the general translation problems.

We will show that the complexity and the difficulty to solve *weak translation problems* can vary considerably.

The complexity and the difficulty of a subproblem π depend on the precise instance of the *weak translation problem* , here denoted

La translittération est aussi un sous-problème de la TA. Elle consiste à surmonter les différences scripturales entre les différents systèmes d'écriture utilisé pour différentes langues [1, 3, 4, 9, 15, 16, 47, 50, 57, 59-61, 65, 73, 82-85, 97, 100, 101, 108, 112, 124, 130, 143-146, 150, 153, 165, 168, 174, 181, 189, 191] ou même pour la même langue [118-121, 164].

Nous nous intéressons à la classe spéciale des sous-problèmes π de TA où N est soit très petit, disons toujours inférieur à 5, voire presque toujours égal à 1 en raison de la proximité des formes écrites de *SL* et *TL*.

Par exemple, cela arrive dans les situations (1) où les langues d'une paire de traduction sont très proches l'une de l'autre, par exemple, bengali -assamais, hindi-marathi, hindi-ourdou, *etc.,* (2) lorsque la traduction est effectuée entre deux variétés ou dialectes d'une langue, que ce soit écrit dans le même système d'écriture (québécois-français, malais-indonésien) ou dans de systèmes d'écriture mutuellement incompréhensibles (Penjabi, sindhi, Seraiki) et (3) lorsque la même langue est écrite dans les différents scripts mutuellement incompréhensibles (cachemiri, malais, penjabi, sindhi, seraiki).

Le domaine de notre recherche est la classe des sous-problèmes π de TA, appliqués à une paire $\langle(L_i, W_j), (L_k W_l)\rangle$ de combinaisons d'une langue et d'un système d'écriture, telle qu'il existe une seule (dans la plupart des cas) ou un ensemble très petit de « solutions de traduction » valides à un sous-problème π, pour une phrase donnée S de L_i écrite en W_j.

Une hypothèse naturelle est que ces problèmes devraient être très simples (en termes de complexité du modèle informatique suffisant) et pas très difficiles (en termes des coûts humains et des coûts de calcul entraînés par la préparation du système pour effectuer la traduction) que les problèmes de traduction générale.

Nous allons montrer que la complexité et la difficulté de résolution des *problèmes faibles de traduction* peuvent varier considérablement.

La complexité et la difficulté d'un sous-problème π dépendent de l'instance précise du *problème de traduction faible,* ici représenté

by $\pi\ \langle(L_i, W_j), (L_k W_l)\rangle$.

We will also use the notation $\pi(SL/SW, TL/TW$.

For example, the complexity and difficulty of interdialectal translation is less for Malay/Latin–Indonesian/Latin than for Hindi/Devanagari-h–Marathi/Devanagari-m translation[50].

We can categorize *weak translation problems* into generic and specific subproblems.

Intralingual localization is a generic problem that can be further refined in the specific problems of *word for word translation* and *intralingual translation* between different varieties of the same language.

For example, IBM product documentation in French is translated into French by Bull[51], a French computer company that sells IBM products (eg AS4000 under AIX) as OEM.

Bull does not use the French versions prepared by IBM because IBM terminology is not identical to Bull terminology[52].

This kind of translation is also mandatory to localize the Quebecois dialect in France, *e.g.* the Quebecois term '*présentement*' must be localized into '*maintenant*' in France and vice versa.

Similar problems also exist between English (UK) & English (USA), French of 14 th century–standard French, and Malay (Malaysia)–Indonesian (Indonesia).

To solve these problems, a full syntactic anal-

par $\pi\ \langle(L_i, W_j), (L_k W_l)\rangle$.

Nous utiliserons également la notation $\pi(SL/SW, TL/TW$.

Par exemple, la complexité et la difficulté de la traduction interdialectale est moindre pour le malais/latin-indonésien/latin que pour la traduction hindi/devanagari-h-marathi/devanagari-m.

Nous pouvons classer les *problèmes faibles de traduction* en sous-problèmes génériques et spécifiques.

La *localisation intralinguale* est un problème générique qui peut être affinée par les problèmes spécifiques de la *traduction mot à mot* et la *traduction intralinguale* entre les différentes variétés de la même langue.

Par exemple, la documentation de produits IBM en français est traduite en français par Bull, une société informatique française qui vend des produits IBM (par exemple, sous AIX AS4000) en OEM.

Bull n'utilise pas les versions françaises préparées par IBM, car la terminologie IBM n'est pas identique à la terminologie de Bull.

Ce type de traduction est également obligatoire pour localiser le dialecte québécois en français, par exemple le terme québécois *« présentement »* doit être localisé en *« maintenant »* en France et vice-versa.

Des problèmes similaires existent également entre l'anglais (Royaume-Uni) et l'américain (USA), le français du 14ème siècle et le français standard, et le malais (Malaisie) et l'indonésien (Indonésie).

Pour résoudre ces problèmes, une analyse syn-

[50] The script used for Hindi and Marathi are different variants of the Devanagari script (the original is used for Sanskrit).

Les script utilisés pour le hindi et le marathi sont différentes variantes de l'alphabet devanagari (l'original est utilisé pour le sanskrit).

[51] We refer here to the documentation of AIX, IBM proprietary version of UNIX http://www.bull.com/index.php.

Nous nous référons ici à la documentation de AIX, IBM version propriétaire de UNIX http://www.bull.com/index.php

[52] As Hagège said, "languages are the flags of national identity". Here, company terminologies are flags of company identities.

Comme Hagège l'a dit: « les langues sont les drapeaux des identités nationales ». Ici, les terminologies des compagnies sont des drapeaux de l'identité des compagnies.

ysis is not required, but we have to perform a *word for word translation* for the localization of one variety or dialect of the language into the other and vice versa.

Table 1 gives a list of generic and specific subproblems of the general *weak translation problem* , together with some of their instances, in increasing complexity and difficulty order.

Successive generic problems are more complex than the previous ones.

For example, the Quebecois–French pair relates to both the first and the third generic problem.

In case of intralingual localization, *word for word translation* is sufficient to perform lexical Quebecois–French translation, but we need to do a more complex analysis to perform interdialectal Quebecois–French translation.

taxique complète n'est pas nécessaire, mais nous devons effectuer une *traduction mot à mot* pour la localisation d'une variante ou d'un dialecte de la langue dans un autre et vice versa.

La Table 1 donne une liste de sous-problèmes spécifiques et génériques du *problème faible de traduction* général, ainsi que certaines de leurs instances, en ordre de complexité et de difficulté croissante.

Les problèmes génériques successifs sont plus complexes que les précédents.

Par exemple, la paire français-québécois concerne à la fois le premier et le troisième problème générique.

En cas de localisation intralinguale, la *traduction mot à mot* est suffisante pour effectuer la traduction lexicale québécois-français, mais nous avons besoin d'effectuer une analyse plus complexe pour effectuer une traduction interdialectale français-québécois.

Sr.	Generic Subproblem Sous-problème générique	Specific Subproblems Sous-problèmes spécifiques	Instances Instances	Constraints Contraintes
1	Language localization localisation linguistique or ou intralingual localization localisation intralinguale	Word for word translation Traduction mot à mot Intralingual translation traduction intralinguale	Unix documentation IBM to Bull (French to French) Documentation Unix d'IBM vers Bull (français vers français)	$SL = TL$ $SW = TW$
			Québécois–French Québécois-français	
			Malay–Indonesian Malais-indonésien	
2	Scriptural translation Traduction scripturale	Transliteration Translitération Transcription Transcription	Malay/Latin–Malay/Jawi Malais/latin-malais/jawi	$SL = TL$ $SW \neq TW$
			Sindhi/Sindhi[53] - Sindhi/Devanagari Sindhi/sindhi	

[53] The derivation of the Perso-Arabic script is known as the Sindhi script.

La dérivation de l'écriture arabo-persane est connue sous le nom de script sindhi.

			–Sindhi/devanagari	
		Phonetic transcription Transcription phonétique	Punjabi/Gurmukhi–Punjabi/Shahmukhi Punjabi/gurmukhi-Punjabi/shahmukhi	
			French/Roman–French/IPA[54] Français/roman-français/IPA	
		Transliteration Translitération	Hindi–Urdu Hindi-ourdou	$SL \neq TL$ $SW \neq TW$
		Transcription Transcription	Bengali–Assamese Bengali-assamais	
		Phonetic transcription Transcription phonétique	Hindi–Marathi Hindi-marathi	
3	Interdialectal translation traduction interdialectale	Word for word translation Traduction mot à mot	Quebecois–French Québécois-français	$SL = TL$ $SW = TW$
			English (USA)–English (UK) Anglais (USA)-anglais (UK)	
			Malay/Latin–Indonesian/Latin Malais/latin-indonésien/latin	
		Scriptural translation Traduction scripturale	Sindhi/Sindhi–Sindhi/Devanagari Sindhi/sindhi sindhi/devanagari	$SL = TL$ $SW \neq TW$
		Intralingual translation traduction intralinguale	Punjabi/Gurmukhi–Punjabi/Shahmukhi Punjabi/gurmukhi-Punjabi/shahmukhi	
			Malay/Jawi–Indonesian/Latin Malais/jawi-indonésien/latin	
4	Bilingual translation Traduction bilingue	Word for word translation	Hindi–Urdu Hindi-ourdou	$SL \neq TL$ $SW \neq TW$

[54] International Phonetic Alphabet (IPA). | Alphabet phonétique international (API).

		Traduction mot à mot	Bengali–Assamese Bengali-assamais	
		Scriptural translation Traduction scripturale		
		Bilingual translation between linearly very similar languages Traduction bilingue entre des langues linéairement très semblables	Hindi–Marathi Hindi-marathi	

Table 1: Subproblems of the weak translation problem (by order of increasing complexity)

Table 1 : Sous-problèmes du problème faible de traduction (par ordre croissant de complexité)

Linguistic Architecture

Following [21, 26, 28, 178, 179, 190], we have adopted and adapted the *framework for syntactic translation* , shown in Figure, to solve the *weak translation problems*.

Architecture linguistique

À la suite de [21, 26, 28, 178, 179, 190], nous avons adopté et adapté le *cadre pour la traduction syntaxique,* montré dans la Figure 1, pour résoudre les *problèmes faibles de traduction.*

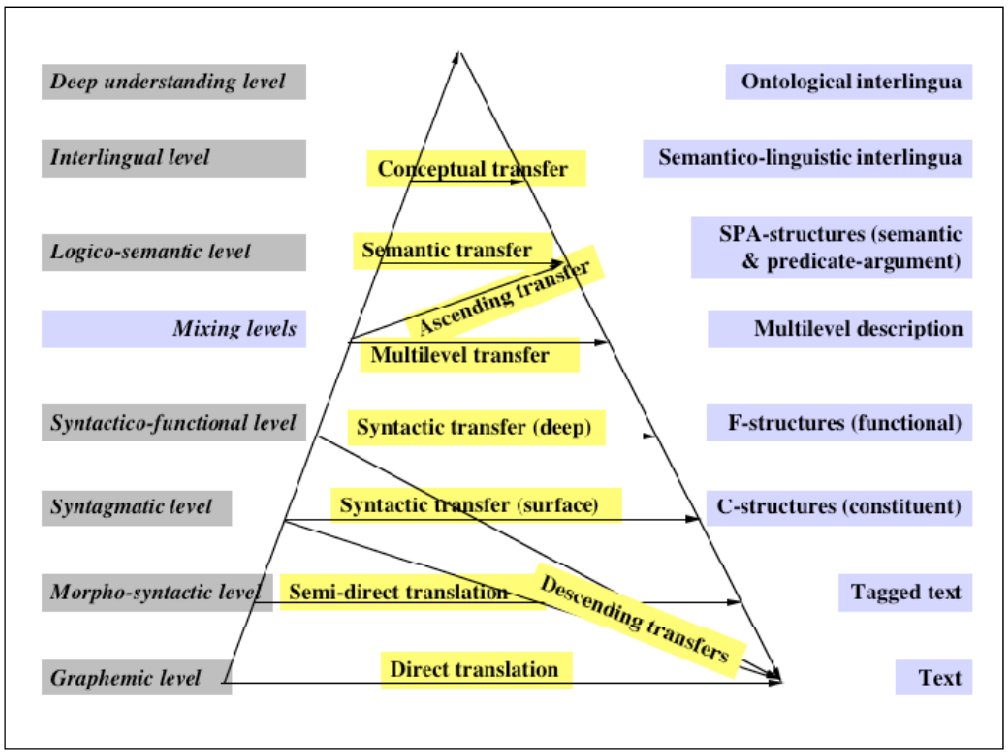


Figure 1: *Vauquois' triangle [21, 26, 28, 178, 179]*

Figure 1: *Vauquois triangle [21, 26, 28, 178, 179]*

We use *interlingua* and *transfer-based* linguistic architectures, and experiment with expert, empirical and hybrid approaches.

We go through various levels of linguistic representation: source (in SW), morphotactic, morphological, morphosyntactic and Universal Intermediate Transcription (UIT) .

The refinement of Vauquois' triangle to the weak translation problems is shown in Figure 2.

Nous employons des architectures linguistiques à *interlingua* et à *transfert*, et expérimentons avec des approches expertes, empiriques et hybrides.

Nous passons par divers niveaux de représentation linguistique : source (dans le système d'écriture source SW), morphotactique, morphologique, morphosyntaxique, et transcription intermédiaire universelle (UIT).

Le raffinement du "triangle de Vauquois" aux problèmes faibles de traduction est montré dans la Figure 2.

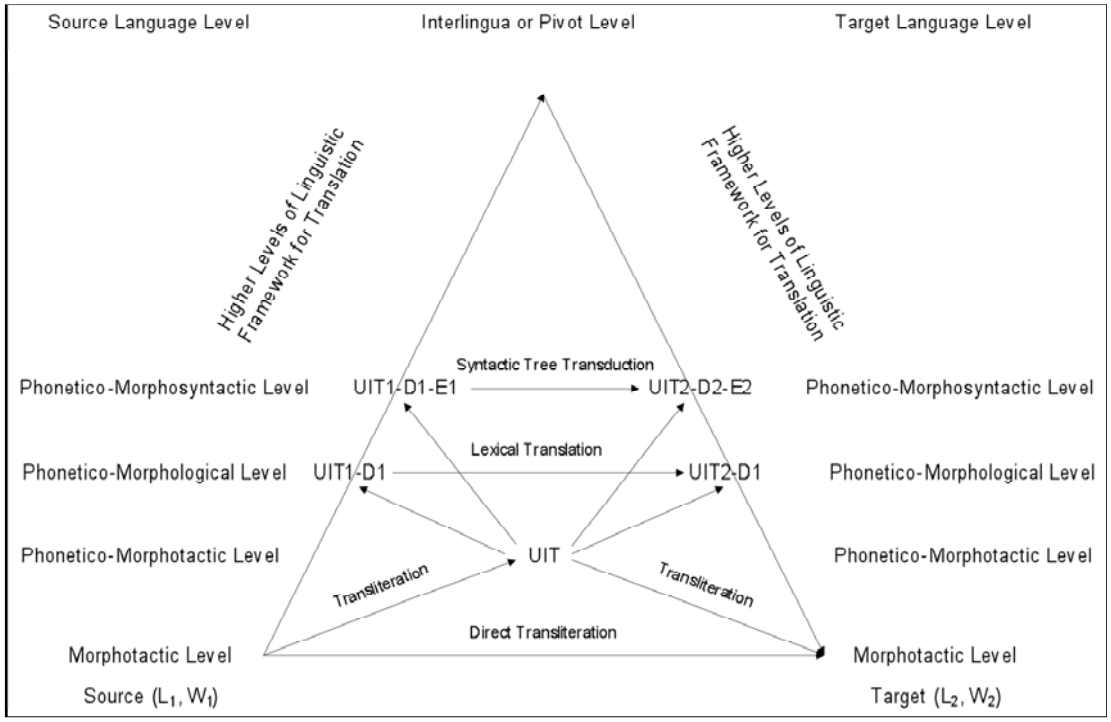


Figure 2*: Adopted and adapted Vauquois's triangle for the weak translation problem*

Figure 2: le triangle de Vauquois adopté et adapté au problème faible de traduction

UIT is defined for each group of very closely related languages or dialects and serves as a Pivot.

More precisely, it is used as a *phonetico-morphotactic* pivot for surface morphotactic translation, as a *phonetico-morphological* pivot for word for word translation, and as a *phonetico-morphosyntactic* lexical pivot for syntax-based translation (in conjunction with syntactic transfer).

L'UIT est définie pour chaque groupe de langues apparentées de très près ou de dialectes et sert de pivot.

Plus précisément, il est employé comme un pivot *phonético-morphotactique* pour la traduction morphotactique de surface, comme pivot *phonético-morphologique* pour la traduction mot à mot, et comme pivot *phonético-morphosyntactique* pour la traduction basée sur la syntaxe (en conjonction avec le transfert syntaxique).

Computational Model

Researchers have employed various computational models ranging from *finite-state technology* [131-135, 161, 162, 166, 167, 169] to *machine learning* , and *empirical methods* [1, 32, 33, 41, 102, 104, 115] for solving different subproblems of MT.

Following [3, 4, 100, 101, 134, 135, 162, 174], we employ *finite-state technology* for solving different subproblems of the weak translation problem.

For example, we use non-probabilistic finite-state transducers [121] to solve the problem of *scriptural translation* (we will define that term precisely later).

We also use *Context-Free Grammar* (CFG) for developing "phrase structure grammars".

Finite-state methods give a 16.1% *word error rate* for Urdu to Hindi *scriptural translation* when all necessary information is present in the input text (we will explain later what we mean by necessary information).

They give a 47% *word error rate* when all necessary information is not present in the input text (the usual and normal case especially for Urdu to Hindi, Punjabi/Shahmukhi to Punjabi/Gurmukhi, *etc* .).

At sentence level, the finite-state methods give a 90% *sentence error rate* for Urdu to Hindi scriptural translation when the input Urdu text contains the required information.

Without the required information in the input Urdu text, they give a 99% sentence error rate.

Due to the successful use of SMT models in MT, we conjectured that SMT could give us better results than our finite-state model.

Indeed, SMT decreases the *word error rate* from 47% to 22.1% for Urdu to Hindi transliteration when the input Urdu text does not con-

Modèle de calcul

Les chercheurs ont utilisé différents modèles de calcul à partir de *la technologie à états finis* [131-135, p. 161, 162, 166, 167, 169] pour *l'apprentissage machine* et des *méthodes empiriques* [1, 32, 33, 41 102, 104, 115] pour résoudre les différents sous-problèmes de MT.

À la suite de [3, 4, 100, 101, 134, 135, 162, 174], nous employons *la technologie à états finis* pour la résolution de sous-problèmes différents du problème faible de traduction.

Par exemple, nous utilisons des transducteurs d'états finis non-probabilistes [121] pour résoudre le problème de la *traduction scripturale* (nous définirons précisément ce terme plus tard).

Nous utilisons aussi des *grammaires hors-contexte* (CFG) pour le développement de « grammaires syntagmatiques ».

Les méthodes à états finis donnent un *taux d'erreur en mots* de 16,1% pour la *traduction scripturale* ourdou-hindi quand toute l'information nécessaire est présente dans le texte d'entrée (nous expliquerons plus tard ce que nous voulons dire par « information nécessaire »).

Elles donnent un *taux d'erreur en mots* de 47% quand tous les renseignements nécessaires ne sont pas présents dans le texte d'entrée (cas habituel et normal, notamment pour ourdou-hindi, punjabi/shahmukhi-punjabi/gurmukhi, *etc.)*.

Au niveau des phrases, les méthodes à états finis donnent un taux d'erreur de *90% en phrases* pour la traduction scripturale ourdou-hindi quand le texte ourdou en entrée contient l'information requise.

Sans information requise dans le texte ourdou en entrée, elles donnent un taux d'erreur en phrases de 99%.

En raison de l'utilisation réussie des modèles SMT en TA, nous avons conjecturé que la SMT pourrait nous donner de meilleurs résultats que notre modèle à états finis.

En effet, la SMT diminue le taux d'erreur en *mots* de 47% à 22,1% pour la translittération ourdou-hindi quand le texte ourdou en entrée

tain the diacritical marks (mandatory for performing Urdu to Hindi *scriptural translation*).

ne contient pas les marques diacritiques (nécessaires pour effectuer une bonne *traduction scripturale* ourdou-hindi).

At sentence level, it decreases the error rate from 99% to 95%.

Au niveau de la phrase, elle fait diminuer le taux d'erreur de 99% à 95%.

In contrast, our finite-state model gives better results (16.1% word error rate) than our best SMT model (27.8% word error rate) when the Urdu text does contain all necessary diacritical marks.

En revanche, notre modèle à états finis donne de meilleurs résultats (16,1% de taux d'erreur en mots) que notre meilleur modèle SMT (27,8% de taux d'erreur en mots) lorsque le texte en ourdou contient toutes les marques diacritiques nécessaires.

The absence of information in the source side, which is the usual case for Urdu to Hindi scriptural translation, is a big challenge and cannot be handled well within the framework of *non-probabilistic finite-state transducers* .

L'absence d'information du côté source, qui est le cas habituel pour la traduction scripturale ourdou-hindi, est un défi important, et ne peut pas être traitée tout à fait bien dans le cadre des *transducteurs d'états-finis non-probabilistes*.

Although SMT increases the word accuracy in such cases, the results are still not satisfactory as far as usability in real context is considered.

Bien que la SMT augmente la précision en mots dans de tels cas, les résultats ne sont toujours pas satisfaisants dans la mesure où la qualité d'usage dans un contexte réel est prise en considération.

To increase the accuracy, we have proposed a *hybrid model* [120] for *scriptural translation* and gained an overall accuracy of 79.1% (word-level) when the input Urdu text does not contain the diacritical marks.

Pour augmenter la précision, nous avons proposé un *modèle hybride* [120] pour la *traduction scripturale* et obtenu une précision globale de 79,1% (au niveau des mots) lorsque le texte ourdou en entrée ne contient pas les signes diacritiques.

A hybrid model, a combination of finite-state and statistical models, gives better results than the previous two models.

Un modèle hybride, combinaison des modèles à états finis et statistiques, donne de meilleurs résultats que les deux modèles précédents.

In short, we have employed finite-state, empirical and statistical, context-fee grammars, tree transduction and syntax-based translation (in conjunction with syntactic transfer) to solve different generic and specific subproblems of *weak translation problems*.

En bref, nous avons employé des modèles à états finis, empiriques et statistiques, des grammaires hors-contexte, de la transduction d'arbres et de la traduction basée sur la syntaxe (en conjonction avec le transfert syntaxique) pour résoudre différents sous-problèmes génériques et spécifiques des problèmes *faibles* de traduction.

Table 2 shows results of Urdu to Hindi scriptural translation of different approaches used to solve the problem of Urdu to Hindi scriptural translation on the same test set.

La Table 2 montre les résultats de la traduction scripturale ourdou-hindi des différentes approches utilisées pour résoudre le problème de la traduction scripturale hindi-ourdou sur le même jeu de test.

Approach Approche	Word Error Rate Taux d'erreur en mots		Sentence Error Rate Taux d'erreur en phrases	
	With information *Avec l'information*	*Without information* *Sans l'information*	*With information* *Avec l'information*	*Without information* *Sans l'information*
FST	16.1%	47%	90%	99%
SMT	27.8%	23%	94,5%	95%
Hybrid Hybride	14,2%	20,9%	86%	93%

Table 2: *Results of Urdu toHindi scriptural translation*

Table 2: Résultats de traduction scripturale ourdou-hindi

Evaluation Methods

One of the most difficult things in MT is the evaluation of a proposed system/algorithm.

A *natural language* is not an object of exact science like Mathematics or Physics.

Therefore, the understanding of a natural language is a subjective problem that depends on multiple factors.

For example, *multivalence* makes it hard to associate a real objective number to an MT evaluation.

Recent MT evaluation campaigns have been criticized because only tables of figures (such as BLEU, NIST, ORANGE, METEOR…) are shown as results, while these n-gram based measures have been shown not to correlate very well with human judgments [40].

Commercial MT systems have been consistently ranked low by these measures, while human judges ranked them quite high [81].

We also have achieved an average 80% word accuracy for Hindi-Urdu scriptural translation with our finite-state methods that seem to be a good measure.

But if we measure the accuracy at the sentence level, then we have an accuracy of 1% to 14%.

Thus, it is important to do subjective evalua-

Méthodes d'évaluation

Une des choses les plus difficiles en TA est l'évaluation d'un système/algorithme proposé.

Une *langue naturelle* n'est pas un objet d'étude d'une science exacte comme les mathématiques ou la physique.

Par conséquent, la compréhension d'une langue naturelle est un problème subjectif qui dépend de multiples facteurs.

Par exemple, la *multivalence* rend difficile d'associer un vrai nombre objectif à une évaluation de la TA.

Les campagnes d'évaluation de TA récentes ont été critiquées parce que seuls des tableaux de chiffres (par exemple, BLEU, NIST, ORANGE, METEOR ...) sont présentés comme résultats, alors qu'on a montré que ces mesures basées sur des n-grammes n'ont pas de bonne corrélation avec les jugements humains [40].

Les systèmes commerciaux de MT ont toujours été classés au bas de ces mesures, tandis que les juges de l'homme les a classées très élevé [81].

Nous avons également obtenu une exactitude moyenne en mots de 80% pour la traduction scripturale Hindi-Urdu avec nos méthodes à états finis, et cette mesure semble être une bonne mesure.

Mais si nous mesurons l'exactitude au niveau des phrases, alors nous avons une exactitude de 1% à 14%.

Ainsi, il est important de faire des évaluations

tions in addition to the objective evaluations.

In general, *human* and *machine* (automatic) evaluation methods are used to evaluate an MT system.

Human evaluations of MT judge various aspects of translations, including adequacy, fidelity and fluency [72, 185].

They are relevant, but costly in terms of time and money [72, 147].

For automatic evaluations, there exist different evaluation metrics like BLEU [147], NIST [49], F-measure [177], METEOR [111], *Word Error Rate* (WER) [138], and MaxSim [43]. Finally, there are objective task-based metrics measuring human performance like post-editing time and time to goal (eg time to perform a booking, a spoken dialogue translation system).

Mostly, we have used n-gram based *automatic* evaluation methods, but also have used *subjective human* evaluation while the post editing for achieving a perfect result.

We have evaluated the quality of our results on different aspects like *Sentence Accuracy* (SA), *Word Accuracy* (WA), *post-editing time* , *confidence level of human evaluator* ,*fluency* , *adequacy* and *usability* .

We have devised scales for each aspect used for categorizing different systems and measuring their translation quality.

For example, the scale devised for SAR is formulated in Table 3. We will discuss and formulate all 228 étai scales later in more 228étails in the sequel.

subjectives en plus des évaluations objectives.

En général, les méthodes d'évaluation *humaines* et *machine* (automatique) sont utilisées pour évaluer un système de TA.

Les évaluations humaines de la TA jugent divers aspects de la traduction, y compris l'adéquation, la fidélité et la fluidité [72, 185].

Elles sont pertinentes, mais coûteuses en termes de temps et d'argent [72, 147].

Pour les évaluations automatiques, il existe des mesures d'évaluation différentes, comme BLEU [147], NIST [49], la F-mesure [177], METEOR [111], l'erreur en mots, *Word Error Rate* (WER) [138], et MaxSim [43]. Enfin, il y a des métriques objectives finalisées (liées à la tâche), mesurant la performance humaine, comme le temps de post-édition et le temps pour atteindre un objectif (par exemple le temps pour effectuer une réservation, pour un système de traduction de dialogues parlés).

Nous avons principalement employé des méthodes d'évaluation *automatiques* basées sur des n-grammes, mais nous avons également employé l'évaluation *subjective humaine* durant la post-édition pour obtenir un résultat parfait.

Nous avons évalué la qualité de nos résultats sur différents aspects comme *l'exactitude en phrases* (SA), *l'exactitude en mots* (WA), le *temps de post-édition*, le niveau de *confiance de l'évaluateur humain*, la *fluidité*, *l'adéquation* et l'*utilisabilité*.

Nous avons mis au point des échelles pour chaque aspect utilisé pour classer les différents systèmes et mesurer leur qualité de traduction.

Par exemple, l'échelle conçue pour le SAR est formulée dans la Table 3. Nous allons discuter et formuler toutes ces échelles plus tard en plus de détails dans la suite.

Scale Point Point d'échelle	Relation with SAR Relation avec les SAR	Description Description
0	SAR < 5%	NULL NULL
1	5% ≤ SAR < 10%	OK OK
2	10% ≤ SAR < 15%	AVERAGE MOYEN
3	15% ≤ SAR < 25%	GOOD ENOUGH ASSEZ BON
4	25% ≤ SAR < 50%	GOOD BON
5	50% ≤ SAR ≥ 70%	VERY GOOD TRES BON
6	SAR > 70%	EXCELLENT EXCELLENT

Table 3: Scale based on sentence accuracy for scriptural translation quality

Table 3: Échelle basée sur l'exactitude en phrases pour la qualité scripturale de traduction

N-gram co-occurrence automatic scoring metrics like BLEU and NIST are widely used as benchmark for MT system evaluation especially BLEU, even with its known shortcomings for evaluation of general MT systems [40].

Des métriques automatiques fondées sur les co-occurrences de N-grammes comme BLEU et NIST sont largement utilisées comme étalons pour l'évaluation de systèmes de TA, en particulier BLEU, même avec ses défauts connus par rapport à l'évaluation des systèmes de TA généralistes [40].

We will show that these deficiencies are not that significant in the case of the evaluation of *weak translation problems* , because we have a unique or very small number of references, say 2 to 4.

Nous allons montrer que ces lacunes ne sont pas si importantes dans le cas de l'évaluation des *problèmes faibles de traduction,* parce que nous avons une seule ou un nombre très petit de références, par exemple 2 à 4.

Thus these metrics are good measures for the translation quality of *weak translation problems* .

Ainsi, ces mesures sont de bonnes mesures pour la qualité de la traduction des *problèmes faibles de traduction.*

On top of WER and SER, we have also used BLEU and NIST to evaluate our translation systems.

En sus de WER et SER, nous avons employé BLEU et NIST pour évaluer nos systèmes de traduction.

Thesis Plan

Plan de la thèse

This report is divided into three main parts.

Ce rapport est divisé en trois parties principales.

The first part introduces the weak translation problems.

La première partie présente les problèmes faibles de traduction.

The first chapter introduces and gives an analysis of scriptural translation problem.

Le premier chapitre introduit le problème de la traduction scripturale et en donne une analyse.

In the second chapter, we describe the finite-state approach for solving scriptural translation problems. Finally, we report the results of our finite-state approach on the Indo-Pak languages (the term is explained later).

Dans le deuxième chapitre, nous décrivons l'approche à états finis pour résoudre des problèmes de traduction scripturale. En conclusion, nous présentons les résultats de notre approche à états finis sur les langues Indo-Pak (le terme est expliqué plus tard).

The second part also consists of two chapters.

In the third chapter, we describe our experiments for solving Hindi–Urdu scriptural translation problem using Statistical Machine Translation (SMT) and report our results.

The last part of this chapter, we illustrate our hybrid approach (a novel combination of finite-state and statistical approaches) for solving the scriptural translation problems for the Indo-Pak languages.

In the fourth chapter, we describe our interactive scriptural translation model.

We also describe our evaluation methodologies for the interactive scriptural translation systems.

The third and final part consists of an analysis of *interdialectal machine translation* , a higher-level weak translation problem that requires more complex analysis and computation approaches than the scriptural translation problems.

In this section, we analyze different computation approaches and required resources for solving the interdialectal translation problem for the Indo-Pak languages.

Finally, we conclude our work and give future perspectives of our study.

Conclusion and Perspectives

MT history is more than half a century old.

The field of MT started to emerge with the advent of computer systems.

In the beginning, it was used to translate Russian texts into English for defense and intelligence purposes.

It was a very hot research area in the past and it is also a very central and highly funded area in the present.

The need of MT is increasing day by day because the world is becoming a global village.

In general, a given sentence of n words in the source language may have an exponential number of valid translations, say $N = k^n$ for some k.

A *weak translation problem* is a translation problem for which N is very small, say less than 5 or almost always 1.

In this study, we restricted our scope to the study and analysis of *weak translation problems* only.

La deuxième partie se compose également de deux chapitres.

Dans le troisième chapitre, nous décrivons nos expériences pour résoudre le problème de la traduction scripturale hindi-ourdou en utilisant la traduction automatique statistique (TAS) et présentons nos résultats.

Dans la dernière partie de ce chapitre, nous illustrons notre approche hybride (une combinaison inédite des approches d'états finis et statistiques) pour résoudre les problèmes de traduction scripturale pour des langues Indo-Pak.

Dans le quatrième chapitre, nous décrivons notre modèle interactif de traduction scripturale.

Nous décrivons également nos méthodes d'évaluation des systèmes de traduction interactive scripturale.

La troisième et dernière partie consiste en une analyse de la *traduction automatique interdialectale,* un problème faible de traduction de plus haut niveau qui nécessite des approches plus problèmes à l'analyse et au calcul que les problèmes de traduction scripturale.

Dans cette section, nous analysons différentes approches computationnelles et les ressources requises pour résoudre le problème de la traduction interdialectale pour les langues Indo-Pak.

Enfin, nous concluons notre travail et donnons des perspectives d'avenir de notre étude.

Conclusion et perspectives

L'histoire de la TA a plus d'un demi-siècle.

Le domaine de la TA a commencé à émerger avec l'avènement des systèmes informatiques.

Au début, la TA a été utilisée pour traduire des textes russes en anglais pour la défense et le renseignement.

C'était un domaine de recherche très intense dans le passé et c'est aussi un domaine très central et bénéficiant d'un financement actuellement.

La nécessité de la TA augmente de jour en jour parce que le monde devient un village planétaire.

En général, une phrase donnée de n mots dans la langue source peut avoir un nombre exponentiel de traductions valides, par exemple $N = k^n$ pour un certain k.

Un *problème de traduction faible* est un problème de traduction pour lequel N est très faible, inférieur à 5 ou presque toujours égal à 1.

Dans cette étude, nous avons limité notre champ à l''étude et à l'analyse des *problèmes faibles de traduction* seulement.

We adopted a step-by-step approach to deal with weak translation problems.

We started with the scriptural translation problem that seemed to be a computationally less hard and relatively simple problem.

In the current literature, the terms transliteration and transcription are often confused. We have defined and proposed the new term 'scriptural translation' to denote a combined process of transliteration and/or transcription.

We were optimistic at the start that we would not only be able to solve the scriptural translation problem, but also the much harder and complex problems of interdialectal translation and of translation between closely related languages.

Actually, the study and analysis of the scriptural translation problem proved to be much more complex and hard than our preliminary estimates.

We have experimented with finite-state, statistical and hybrid models to solve the scriptural translation problems.

The graph of Figure 3 shows a brief comparison of results obtained on HU Test Set 2 using these three different models for the Urdu to Hindi scriptural translation.

Nous avons adopté une approche étape par étape pour traiter les problèmes faibles de traduction.

Nous avons commencé par le problème de traduction scripturale, qui semblait être un problème computationnellement moins dur et relativement simple.

Dans la littérature actuelle, les termes de translittération et de transcription sont souvent confondus. Nous avons défini et proposé le nouveau terme de « traduction scripturale » pour désigner un processus combiné de translittération et / ou de transcription.

Nous étions optimistes au début et pensions que nous pourrions non seulement résoudre les problèmes de traduction scripturale, mais également les problèmes beaucoup plus durs et complexes de traduction interdialectale et de traduction entre des langues très proches.

En fait, l'étude et l'analyse du problème de traduction scripturale se sont avérées beaucoup plus complexes et difficiles que nos estimations préliminaires.

Nous avons expérimenté des modèles à états finis, statistiques et hybrides pour résoudre les problèmes de traduction scripturale.

Le graphique de la Figure 3montre une brève comparaison des résultats obtenus sur le Test Set 2 HU à l'aide de ces trois modèles différents, pour la traduction scripturale de hindi en ourdou.

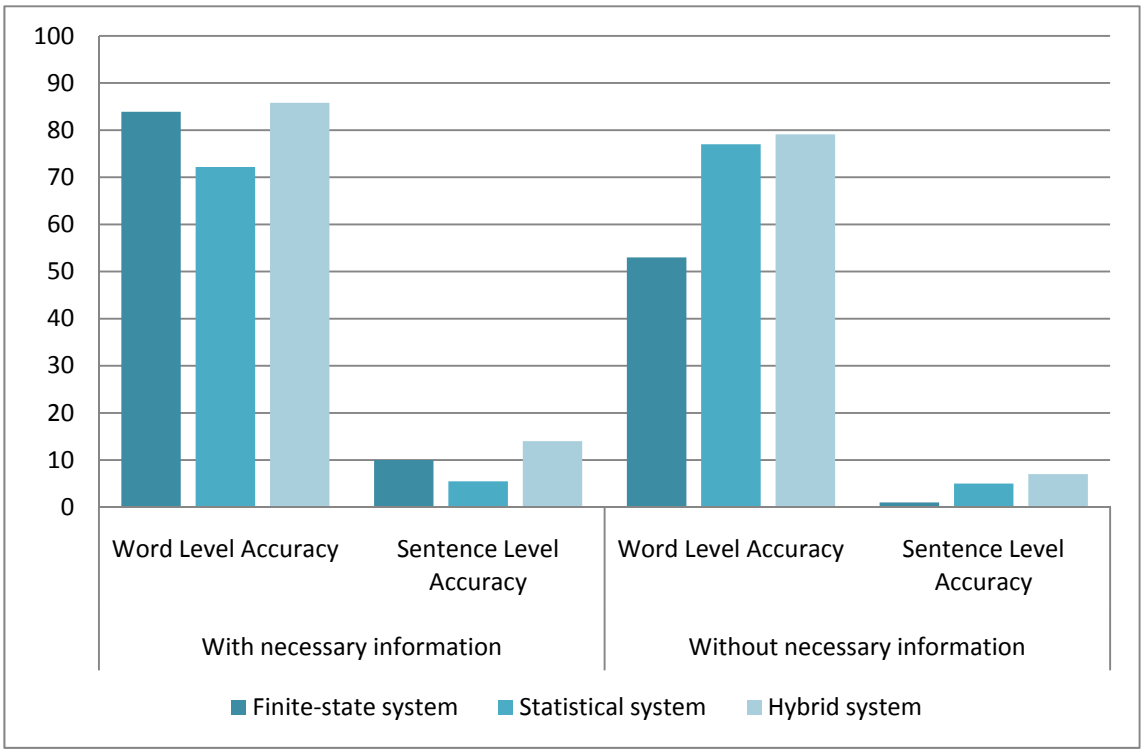


Figure 3: Comparison of Urdu to Hindi scriptural translation systems at word and sentence level

Figure 3 : Comparaison de systèmes de traduction scripturale d'ourdou en hindi aux niveaux des mots et des phrases

The analysis of our scriptural translation results shows that subjective evaluations like user satisfaction, usability of translation result in real life, fluency and adequacy of translated texts are also very important in addition to the objective evaluations like word accuracy, sentence accuracy, and edit-distance.

L'analyse de nos résultats de traduction scripturale montre que les évaluations subjectives, comme la satisfaction des utilisateurs, la facilité d'utilisation des résultats de la traduction dans la vie réelle, la fluidité et l'adéquation des textes traduits, sont également très importantes, en sus des évaluations objectives, comme la précision en mots en phrases, et la distance d'édition.

For subjective evaluation purposes, we have devised different scales to compute different subjective measures for our scriptural translation systems.

Aux fins d'évaluation subjective, nous avons conçu des échelles différentes pour calculer les différentes mesures subjectives de nos systèmes de traduction scripturale.

To get a real life rating for our translation systems and improve the performance of our scriptural translation systems, we have also developed an online interactive scriptural translation system.

Pour obtenir une estimation liée à la vie réelle pour nos systèmes de traduction, et améliorer l'efficacité de nos systèmes de traduction scripturale, nous avons également développé un système de traduction scripturale interactif en ligne.

This interactive scriptural translation system serves different purposes.

Ce système de traduction scripturale interactive sert à différents objectifs.

Although we were not yet able to develop practical interdialectal translation systems, we have presented a theoretical study of how we can develop interdialectal translation systems based on Statistical Machine Translation (SMT) and syntactic translation (based on syntactic transfer and linguistic framework) approaches.

Bien que nous n'ayons pas encore été en mesure de développer des systèmes pratiques de traduction interdialectale, nous avons présenté une étude théorique sur la façon dont nous pouvons développer des systèmes de traduction interdialectale basés sur des approches de traduction automatique statistique (TAS) et de traduction syntaxique (basées sur le transfert syntaxique et sur un cadre linguistique).

That includes a study of what kind of linguistic resources (parallel lexicons and corpora) and lingware modules (morphological and syntactic analyzers) are required for building interdialectal translation systems.

Cela comprend une étude sur le type de ressources linguistiques (lexiques et corpus parallèles) et de modules linguiciels (analyseurs morphologiques et syntaxiques) nécessaires pour la construction de systèmes de traduction interdialectale.

The availability of interactive scriptural translation systems will play a vital role in developing data resources online using the very large Internet community.

La disponibilité de systèmes de traduction scripturale interactifs jouera un rôle essentiel dans le développement de ressources en ligne utilisant la très grande communauté Internet.

We have mainly experimented on Indo-Pak languages, which represent a large population of the world.

Nous avons essentiellement fait des expériences sur des langues Indo-Pak, qui représentent une grande part de la population du monde.

The Hindi–Urdu pair alone represents 1,017 million speakers around the globe.

La paire Ourdou-Hindi représente à elle seule 1.017 millions de locuteurs dans le monde entier.

Only Chinese has more than 1,000 million speakers.

Seul le chinois compte plus de 1.000 millions de locuteurs.

Table 4 shows the number of speakers of IndoPak languages.

Le tableau 4 indique le nombre de locuteurs des langues Indo-Pak.

Sr.	Language	Number of Speakers
1	Hindi	853,000,000
2	Urdu	164,290,000
3	Punjabi	120,000,000
4	Sindhi	21,382,120
5	Seraiki	13,820,000
6	Kashmir	5,640,940
Total		**1178,133,060**

Table 4: Number of speakers of Indo-Pak languages

Tableau 4 : Nombre de locuteurs de langues Indo-Pak

We have made available online our scriptural translation system for Hindi–Urdu[55],

Punjabi/Shahmukhi–Punjabi/Gurmukhi[56],

and Seraiki/Shahmukhi–Seraiki/Devanagari[57].

We will also make available scriptural translation systems for Sindhi/Sindhi–Sindhi/Devanagari and Kashmiri/Urdu–Kashmiri/Devanagari in the near future.

We have presented theoretical studies for the development of interdialectal translation in the third part of this thesis.

In future, we intend to use use this study and develop necessary linguistic resources and lingware modules for developing statistical and syntactic translation systems for Indo-Pak languages.

Table 4 shows the importance of this study in terms of the size of the populations that, we hope, will benefit from our study.

Nous avons mis en ligne notre système de traduction scripturale pour hindi-ourdou,

punjabi / shahmukhi-punjabi / gurmukhi,

et seraiki/shahmukhi-seraiki/devanagari.

Nous allons également mettre à disposition des systèmes de traduction scripturale pour sindhi/sindhi-sindhi/devanagari et cachemiri/ourdou cachemiri/devanagari dans un proche avenir.

Nous avons présenté des études théoriques pour le développement de la traduction interdialectale dans la troisième partie de cette thèse.

À l'avenir, nous avons l'intention d'utiliser utiliser cette étude et de développer les ressources linguistiques et des modules linguiciels nécessaires pour le développement de systèmes de traduction statistique et syntaxique pour les langues Indo-Pak.

Le Table 4 montre l'importance de cette étude en fonction de la taille des populations qui, nous l'espérons, pourront profiter de notre étude.

Extended French Summary

Introduction

En général, le terme *traduction* est compris comme le processus de compréhension du sens d'un texte dans une langue et ensuite de production d'un texte équivalent dans une autre langue, transmettant le même message. La traduction automatique (TA) est un *rêve* des années 1950 [21, 24-26, 80, 125, 171, 172, 178, 179]. Bien qu'un grand nombre d'étapes importantes aient été réalisées pour animer le *rêve* de la TA [21, 23-26, 28, 32, 41, 80, 87-90, 93, 100-102, 104, 105, 113, 115, 125, 131, 132, 134, 135, 139, 144, 162, 171, 172, 178, 179, 188], c'est toujours un rêve dans la recherche interdisciplinaire en *informatique, intelligence artificielle, apprentis-*

[55] http://puran.info/HUMT/index.html

[56] http://puran.info/PMT/index.html

[57] http://puran.info/saraikiMT/index.html

sage automatique, linguistique computationnelle, traitement des langues naturelles, et ingénierie.

Le rêve de la TA est brouillé, comme d'autres rêves. Pour le rendre précis, nous devons considérer des tâches précises de traduction. La TA généraliste, de haute qualité et entièrement automatique est censée être impossible [10, 24-26]. Mais le problème général de la TA peut être réduit à divers sous-problèmes évidemment moins complexes et moins durs que le problème général. Nous nous concentrerons sur quelques-uns d'entre eux, d'un intérêt particulier, tels que la traduction *intralingue* ou *interdialectale.* Cette réduction du problème peut être faite sur la base du domaine de l'application, du sous-langage (une partie restreinte et limitée d'une langue) considéré pour la traduction, des utilisateurs prévus, des couples de langues considérés, *etc.*

Ces fonctionnalités aident également à définir le but et les objectifs des sous-problèmes de la TA [21, 23-27, 41, 80, 93, 102, 115, 125, 139, 171, 172, 178, 179, 187, 190]. Toutefois, ces sous-problèmes peuvent encore être très difficiles et complexes, bien que certaines instances puissent être très simples ou pas aussi difficiles. Nous définirons plus tard avec plus de précision ce que nous voulons dire par complexité et difficulté. Dans tous les cas, il ya un long chemin à parcourir pour progresser [25, 26, 139, 187]. Un de nos objectifs sera de caractériser la complexité et la difficulté de résoudre une certaine classe de problèmes de traduction que nous appellerons *« problèmes faibles de traduction ».*

La TA est connue pour son caractère complexe et la *multivalence* (ambiguïté et polysméie) en est l'une des principales raisons. *Multivalence,* le terme utilisé par Mel'čuk [125], apparaît en raison du non-déterminisme *(polysémie* en cours d'analyse et *synonymie* en génération, et les deux en transfert) [6, 13, 21, 23-26, 32, 41, 51, 93, 102, 113, 115, 125, 139, 149, 154, 178, 179, 187]. Le nombre de traductions possibles d'une phrase moyenne en langue source peut aller jusqu'à des milliers ou en général augmenter de façon spectaculaire avec sa longueur [6, 13, 21, 25, 26, 41, 93, 113, 125, 139, 149, 154, 178, 179, 187]. Une langue source *SL* étant donnée, ainsi qu'une langue cible *TL*, une unité de traduction *S* de *n* mots peut avoir un nombre exponentiel $T_1, T_2, \dots, T_N$ de traductions valides, où $N = O(k^n)$ pour un certain $k > 1$ dépendant du sous-problème précis à traiter.

Pour résoudre le problème de la *multivalence* pour un sous-problème donné de TA, différents filtres sont appliqués à différents niveaux pendant les phases de pré-traitement, d'analyse, de synthèse et de post-traitement pour restreindre la cardinalité de l'ensemble des solutions possibles du problème à un plage de valeurs acceptable et raisonnable [6, 13, 21, 23-28, 41, 51, 80, 93, 102, 104, 113, 115, 125, 139, 149, 154, 171, 172, 178, 187].

La translittération est aussi un sous-problème de la TA. Elle consiste à surmonter les différences scripturales entre les différents systèmes d'écriture utilisé pour différentes langues [1, 3, 4, 9, 15, 16, 47, 50, 57, 59-61, 65, 73, 82-85, 97, 100, 101, 108, 112, 124, 130, 143-146, 150, 153, 165, 168, 174, 181, 189, 191] ou même pour la même langue [118-121, 164].

Nous nous intéressons à la classe spéciale des sous-problèmes π de TA où *N* est soit très petit, disons toujours inférieur à 5, voire presque toujours égal à 1 en raison de la proximité des formes écrites de *SL* et *TL*. Par exemple, cela arrive dans les situations (1) où les langues d'une paire de traduction sont très proches l'une de l'autre, par exemple, bengali -assamais, hindi-marathi, hindi-ourdou, *etc.,* (2) lorsque la traduction est effectuée entre deux variétés ou dialectes d'une langue, que ce soit écrit dans le même système d'écriture (québécois-français, malais-indonésien) ou dans de systèmes d'écriture mutuellement incompréhensibles (Penjabi, sindhi, Seraiki) et (3) lorsque la même langue est écrite dans les différents scripts mutuellement incompréhensibles (cachemiri, malais, penjabi, sindhi, seraiki).

Le domaine de notre recherche est la classe des sous-problèmes π de TA, appliqués à une paire $\langle (L_i, W_j), (L_k W_l) \rangle$ de combinaisons d'une langue et d'un système d'écriture, telle qu'il existe une seule (dans la plupart des cas) ou un ensemble très petit de « solutions de traduction » valides à un sous-problème π, pour une phrase donnée *S* de L_i écrite en W_j. Une hypothèse naturelle est que ces problèmes devraient être très simples (en termes de complexité du modèle in-

formatique suffisant) et pas très difficiles (en termes des coûts humains et des coûts de calcul entraînés par la préparation du système pour effectuer la traduction) que les problèmes de traduction générale. Nous allons montrer que la complexité et la difficulté de résolution des *problèmes faibles de traduction* peuvent varier considérablement.

La complexité et la difficulté d'un sous-problème π dépendent de l'instance précise du *problème de traduction faible,* ici représenté par $\pi\ \langle(L_i, W_j), (L_k W_l)\rangle$. Nous utiliserons également la notation $\pi(SL/SW, TL/TW)$. Par exemple, la complexité et la difficulté de la traduction interdialectale est moindre pour le malais/latin-indonésien/latin que pour la traduction hindi/devanagari-h-marathi/devanagari-m. Nous pouvons classer les *problèmes faibles de traduction* en sous-problèmes génériques et spécifiques.

La *localisation intralinguale* est un problème générique qui peut être affinée par les problèmes spécifiques de la *traduction mot à mot* et la *traduction intralinguale* entre les différentes variétés de la même langue. Par exemple, la documentation de produits IBM en français est traduite en français par Bull, une société informatique française qui vend des produits IBM (par exemple, sous AIX AS4000) en OEM. Bull n'utilise pas les versions françaises préparées par IBM, car la terminologie IBM n'est pas identique à la terminologie de Bull.

Ce type de traduction est également obligatoire pour localiser le dialecte québécois en français, par exemple le terme québécois *« présentement »* doit être localisé en *« maintenant »* en France et vice-versa. Des problèmes similaires existent également entre l'anglais (Royaume-Uni) et l'américain (USA), le français du 14ème siècle et le français standard, et le malais (Malaisie) et l'indonésien (Indonésie). Pour résoudre ces problèmes, une analyse syntaxique complète n'est pas nécessaire, mais nous devons effectuer une *traduction mot à mot* pour la localisation d'une variante ou d'un dialecte de la langue dans un autre et vice versa. La Table 1 donne une liste de sous-problèmes spécifiques et génériques du *problème faible de traduction* général, ainsi que certaines de leurs instances, en ordre de complexité et de difficulté croissante.

Les problèmes génériques successifs sont plus complexes que les précédents. Par exemple, la paire français-québécois concerne à la fois le premier et le troisième problème générique. En cas de localisation intralinguale, la *traduction mot à mot* est suffisante pour effectuer la traduction lexicale québécois-français, mais nous avons besoin d'effectuer une analyse plus complexe pour effectuer une traduction interdialectale français-québécois.

Sr.	Sous-problème générique	Sous-problèmes spécifiques	Instances	Contraintes
1	localisation linguistique ou localisation intralinguale	Traduction mot à mot traduction intralinguale	Documentation Unix d'IBM vers Bull (français vers français)	$SL = TL$ $SW = TW$
			Québécois-français	
			Malais-indonésien	
2	Traduction scripturale	Translitération Transcription Transcription phonétique	Malais/latin-malais/jawi	$SL = TL$ $SW \neq TW$
			Sindhi/sindhi[58] – Sindhi/devanagari	
			Punjabi/gurmukhi-Punjabi/shahmukhi	
			Français/roman-français/API[59]	
		Translitération Transcription Transcription phonétique	Hindi-ourdou	$SL \neq TL$ $SW \neq TW$
			Bengali-assamais	
			Hindi-marathi	
3	traduction interdialectale	Traduction mot à mot Traduction scripturale traduction intralinguale	Québécois-français	$SL = TL$ $SW = TW$
			Anglais (USA)-anglais (UK)	
			Malais/latin-indonésien/latin	
			Sindhi/sindhi sindhi/devanagari	$SL = TL$ $SW \neq TW$
			Punjabi/gurmukhi-Punjabi/shahmukhi	
			Malais/jawi-indonésien/latin	
4	Traduction bilingue	Traduction mot à mot Traduction scripturale Traduction bilingue entre des langues linéairement très semblables	Hindi-ourdou	$SL \neq TL$ $SW \neq TW$
			Bengali-assamais	
			Hindi-marathi	

Table 1 : Sous-problèmes du problème faible de traduction (par ordre croissant de complexité)

Architecture linguistique

À la suite de [21, 26, 28, 178, 179, 190], nous avons adopté et adapté le *cadre pour la traduction syntaxique,* montré dans la Figure 1, pour résoudre les *problèmes faibles de traduction.*

[58] La dérivation de l'écriture arabo-persane est connue sous le nom de script sindhi.

[59] Alphabet phonétique international (API).

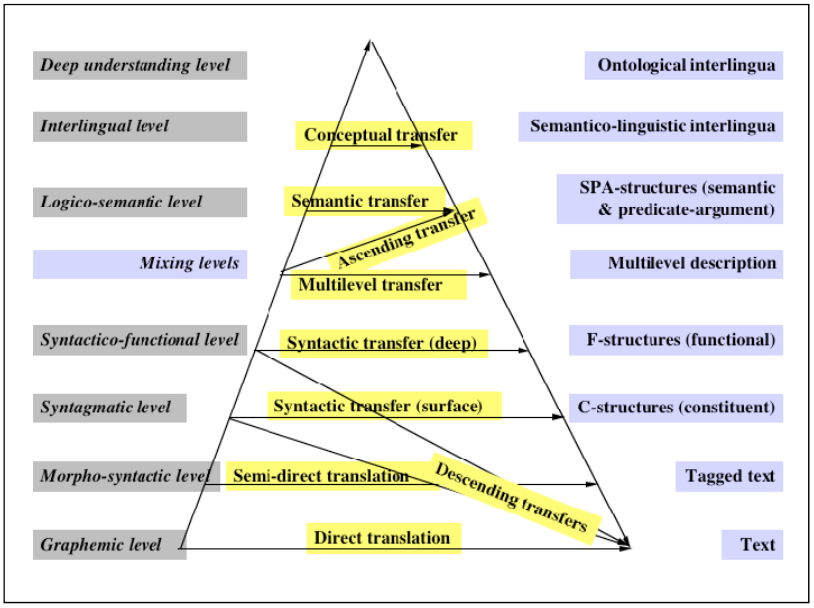


Figure 1: Vauquois triangle [21, 26, 28, 178, 179]

Nous employons des architectures linguistiques à *interlingua* et à *transfert*, et expérimentons avec des approches expertes, empiriques et hybrides. Nous passons par divers niveaux de représentation linguistique : source (dans le système d'écriture source SW), morphotactique, morphologique, morphosyntaxique, et transcription intermédiaire universelle (UIT). Le raffinement du "triangle de Vauquois" aux problèmes faibles de traduction est montré dans la Figure 2.

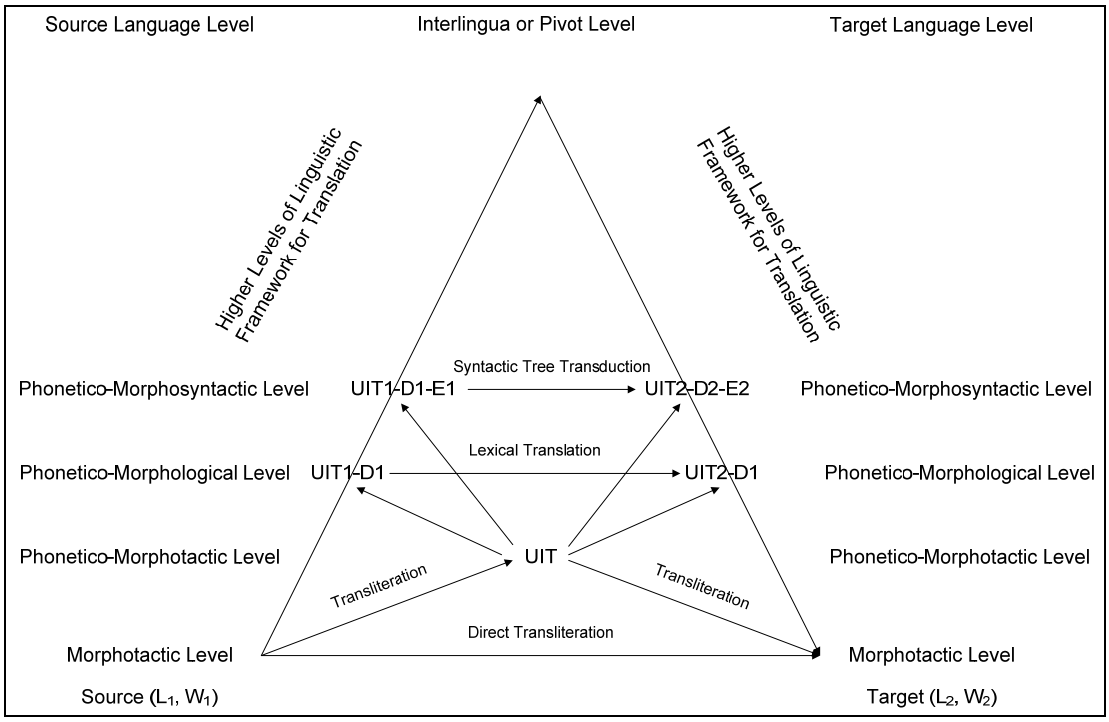


Figure 2: le triangle de Vauquois adopté et adapté au problème faible de traduction

L'UIT est définie pour chaque groupe de langues apparentées de très près ou de dialectes et sert de pivot. Plus précisément, il est employé comme un pivot *phonético-morphotactique* pour la traduction morphotactique de surface, comme pivot *phonético-morphologique* pour la traduction mot à mot, et comme pivot *phonético-morphosyntactique* pour la traduction basée sur la syntaxe (en conjonction avec le transfert syntaxique).

Modèle de calcul

Les chercheurs ont utilisé différents modèles de calcul à partir de *la technologie à états finis* [131-135, p. 161, 162, 166, 167, 169] pour *l'apprentissage machine* et des *méthodes empiriques* [1, 32, 33, 41 102, 104, 115] pour résoudre les différents sous-problèmes de MT.

À la suite de [3, 4, 100, 101, 134, 135, 162, 174], nous employons *la technologie à états finis* pour la résolution de sous-problèmes différents du problème faible de traduction. Par exemple, nous utilisons des transducteurs d'états finis non-probabilistes [121] pour résoudre le problème de la *traduction scripturale* (nous définirons précisément ce terme plus tard). Nous utilisons aussi des *grammaires hors-contexte* (CFG) pour le développement de « grammaires syntagmatiques ».

Les méthodes à états finis donnent un *taux d'erreur en mots* de 16,1% pour la *traduction scripturale* ourdou-hindi quand toute l'information nécessaire est présente dans le texte d'entrée (nous expliquerons plus tard ce que nous voulons dire par « information nécessaire »). Elles donnent un *taux d'erreur en mots* de 47% quand tous les renseignements nécessaires ne sont pas présents dans le texte d'entrée (cas habituel et normal, notamment pour ourdou-hindi, punjabi/shahmukhi-punjabi/gurmukhi, *etc.).* Au niveau des phrases, les méthodes à états finis donnent un taux d'erreur de *90% en phrases* pour la traduction scripturale ourdou-hindi quand le texte ourdou en entrée contient l'information requise. Sans information requise dans le texte ourdou en entrée, elles donnent un taux d'erreur en phrases de 99%.

En raison de l'utilisation réussie des modèles SMT en TA, nous avons conjecturé que la SMT pourrait nous donner de meilleurs résultats que notre modèle à états finis. En effet, la SMT diminue le taux d'erreur en *mots* de 47% à 22,1% pour la translittération ourdou-hindi quand le texte ourdou en entrée ne contient pas les marques diacritiques (nécessaires pour effectuer une bonne *traduction scripturale* ourdou-hindi). Au niveau de la phrase, elle fait diminuer le taux d'erreur de 99% à 95%. En revanche, notre modèle à états finis donne de meilleurs résultats (16,1% de taux d'erreur en mots) que notre meilleur modèle SMT (27,8% de taux d'erreur en mots) lorsque le texte en ourdou contient toutes les marques diacritiques nécessaires.

L'absence d'information du côté source, qui est le cas habituel pour la traduction scripturale ourdou-hindi, est un défi important, et ne peut pas être traitée tout à fait bien dans le cadre des *transducteurs d'états-finis non-probabilistes*. Bien que la SMT augmente la précision en mots dans de tels cas, les résultats ne sont toujours pas satisfaisants dans la mesure où la qualité d'usage dans un contexte réel est prise en considération.

Pour augmenter la précision, nous avons proposé un *modèle hybride* [120] pour la *traduction scripturale* et obtenu une précision globale de 79,1% (au niveau des mots) lorsque le texte ourdou en entrée ne contient pas les signes diacritiques. Un modèle hybride, combinaison des modèles à états finis et statistiques, donne de meilleurs résultats que les deux modèles précédents. En bref, nous avons employé des modèles à états finis, empiriques et statistiques, des grammaires hors-contexte, de la transduction d'arbres et de la traduction basée sur la syntaxe (en conjonction avec le transfert syntaxique) pour résoudre différents sous-problèmes génériques et spécifiques des problèmes *faibles* de traduction. La Table 2 montre les résultats de la traduction scripturale ourdou-hindi des différentes approches utilisées pour résoudre le problème de la traduction scripturale hindi-ourdou sur le même jeu de test.

Approche	Taux d'erreur en mots		Taux d'erreur en phrases	
	Avec l'information	*Sans l'information*	*Avec l'information*	*Sans l'information*
FST	16.1%	47%	90%	99%
SMT	27.8%	23%	94,5%	95%
Hybride	14,2%	20,9%	86%	93%

Table 2: Résultats de traduction scripturale ourdou-hindi

Méthodes d'évaluation

Une des choses les plus difficiles en TA est l'évaluation d'un système/algorithme proposé. Une *langue naturelle* n'est pas un objet d'étude d'une science exacte comme les mathématiques ou la physique. Par conséquent, la compréhension d'une langue naturelle est un problème subjectif qui dépend de multiples facteurs. Par exemple, la *multivalence* rend difficile d'associer un vrai nombre objectif à une évaluation de la TA.

Les campagnes d'évaluation de TA récentes ont été critiquées parce que seuls des tableaux de chiffres (par exemple, BLEU, NIST, ORANGE, METEOR ...) sont présentés comme résultats, alors qu'on a montré que ces mesures basées sur des n-grammes n'ont pas de bonne corrélation avec les jugements humains [40]. Les systèmes commerciaux de MT ont toujours été classés au bas de ces mesures, tandis que les juges de l'homme les a classées très élevé [81]. Nous avons également obtenu une exactitude moyenne en mots de 80% pour la traduction scripturale Hindi-Urdu avec nos méthodes à états finis, et cette mesure semble être une bonne mesure. Mais si nous mesurons l'exactitude au niveau des phrases, alors nous avons une exactitude de 1% à 14%. Ainsi, il est important de faire des évaluations subjectives en plus des évaluations objectives.

En général, les méthodes d'évaluation *humaines* et *machine* (automatique) sont utilisées pour évaluer un système de TA. Les évaluations humaines de la TA jugent divers aspects de la traduction, y compris l'adéquation, la fidélité et la fluidité [72, 185]. Elles sont pertinentes, mais coûteuses en termes de temps et d'argent [72, 147]. Pour les évaluations automatiques, il existe des mesures d'évaluation différentes, comme BLEU [147], NIST [49], la F-mesure [177], METEOR [111], l'erreur en mots, *Word Error Rate* (WER) [138], et MaxSim [43]. Enfin, il y a des métriques objectives finalisées (liées à la tâche), mesurant la performance humaine, comme le temps de post-édition et le temps pour atteindre un objectif (par exemple le temps pour effectuer une réservation, pour un système de traduction de dialogues parlés). Nous avons principalement employé des méthodes d'évaluation *automatiques* basées sur des n-grammes, mais nous avons également employé l'évaluation *subjective humaine* durant la post-édition pour obtenir un résultat parfait.

Nous avons évalué la qualité de nos résultats sur différents aspects comme *l'exactitude en phrases* (SA), *l'exactitude en mots* (WA), le *temps de post-édition*, le niveau de *confiance de l'évaluateur humain*, la *fluidité*, *l'adéquation* et l'*utilisabilité*. Nous avons mis au point des échelles pour chaque aspect utilisé pour classer les différents systèmes et mesurer leur qualité de traduction. Par exemple, l'échelle conçue pour le SAR est formulée dans la Table 3. Nous allons discuter et formuler toutes ces échelles plus tard en plus de détails dans la suite.

Point d'échelle	Relation avec les SAR	Description
0	SAR < 5%	NULL
1	5% ≤ SAR < 10%	OK
2	10% ≤ SAR < 15%	MOYEN
3	15% ≤ SAR < 25%	ASSEZ BON
4	25% ≤ SAR < 50%	BON
5	50% ≤ SAR ≥ 70%	TRES BON
6	SAR > 70%	EXCELLENT

Table 3: Échelle basée sur l'exactitude en phrases pour la qualité scripturale de traduction

Des métriques automatiques fondées sur les co-occurrences de N-grammes comme BLEU et NIST sont largement utilisées comme étalons pour l'évaluation de systèmes de TA, en particulier BLEU, même avec ses défauts connus par rapport à l'évaluation des systèmes de TA généralistes [40]. Nous allons montrer que ces lacunes ne sont pas si importantes dans le cas de l'évaluation des *problèmes faibles de traduction,* parce que nous avons une seule ou un nombre très petit de références, par exemple 2 à 4. Ainsi, ces mesures sont de bonnes mesures pour la qualité de la traduction des *problèmes faibles de traduction.* En sus de WER et SER, nous avons employé BLEU et NIST pour évaluer nos systèmes de traduction.

Plan de la thèse

Ce rapport est divisé en trois parties principales. La première partie présente les problèmes faibles de traduction. Le premier chapitre introduit le problème de la traduction scripturale et en donne une analyse. Dans le deuxième chapitre, nous décrivons l'approche à états finis pour résoudre des problèmes de traduction scripturale. En conclusion, nous présentons les résultats de notre approche à états finis sur les langues Indo-Pak (le terme est expliqué plus tard).

La deuxième partie se compose également de deux chapitres. Dans le troisième chapitre, nous décrivons nos expériences pour résoudre le problème de la traduction scripturale hindi-ourdou en utilisant la traduction automatique statistique (TAS) et présentons nos résultats. Dans la dernière partie de ce chapitre, nous illustrons notre approche hybride (une combinaison inédite des approches d'états finis et statistiques) pour résoudre les problèmes de traduction scripturale pour des langues Indo-Pak. Dans le quatrième chapitre, nous décrivons notre modèle interactif de traduction scripturale. Nous décrivons également nos méthodes d'évaluation des systèmes de traduction interactive scripturale.

La troisième et dernière partie consiste en une analyse de la *traduction automatique interdialectale,* un problème faible de traduction de plus haut niveau qui nécessite des approches plus problèmes à l'analyse et au calcul que les problèmes de traduction scripturale. Dans cette section, nous analysons différentes approches computationnelles et les ressources requises pour résoudre le problème de la traduction interdialectale pour les langues Indo-Pak. Enfin, nous concluons notre travail et donnons des perspectives d'avenir de notre étude.

Conclusion et perspectives

L'histoire de la TA a plus d'un demi-siècle. Le domaine de la TA a commencé à émerger avec l'avènement des systèmes informatiques. Au début, la TA a été utilisée pour traduire des textes russes en anglais pour la défense et le renseignement. C'était un domaine de recherche très intense dans le passé et c'est aussi un domaine très central et bénéficiant d'un financement actuellement. La nécessité de la TA augmente de jour en jour parce que le monde devient un village planétaire.

En général, une phrase donnée de n mots dans la langue source peut avoir un nombre exponentiel de traductions valides, par exemple $N = k^n$ pour un certain k. Un *problème de traduction faible* est un problème de traduction pour lequel N est très faible, inférieur à 5 ou presque toujours égal à 1. Dans cette étude, nous avons limité notre champ à l"étude et à l'analyse des *problèmes faibles de traduction* seulement.

Nous avons adopté une approche étape par étape pour traiter les problèmes faibles de traduction. Nous avons commencé par le problème de traduction scripturale, qui semblait être un problème computationnellement moins dur et relativement simple. Dans la littérature actuelle, les termes de translittération et de transcription sont souvent confondus. Nous avons défini et proposé le nouveau terme de « traduction scripturale » pour désigner un processus combiné de translittération et / ou de transcription.

Nous étions optimistes au début et pensions que nous pourrions non seulement résoudre les problèmes de traduction scripturale, mais également les problèmes beaucoup plus durs et complexes de traduction interdialectale et de traduction entre des langues très proches.

En fait, l'étude et l'analyse du problème de traduction scripturale se sont avérées beaucoup plus complexes et difficiles que nos estimations préliminaires. Nous avons expérimenté des modèles à états finis, statistiques et hybrides pour résoudre les problèmes de traduction scripturale. Le graphique de la 3 montre une brève comparaison des résultats obtenus sur le Test Set 2 HU à l'aide de ces trois modèles différents, pour la traduction scripturale de hindi en ourdou.

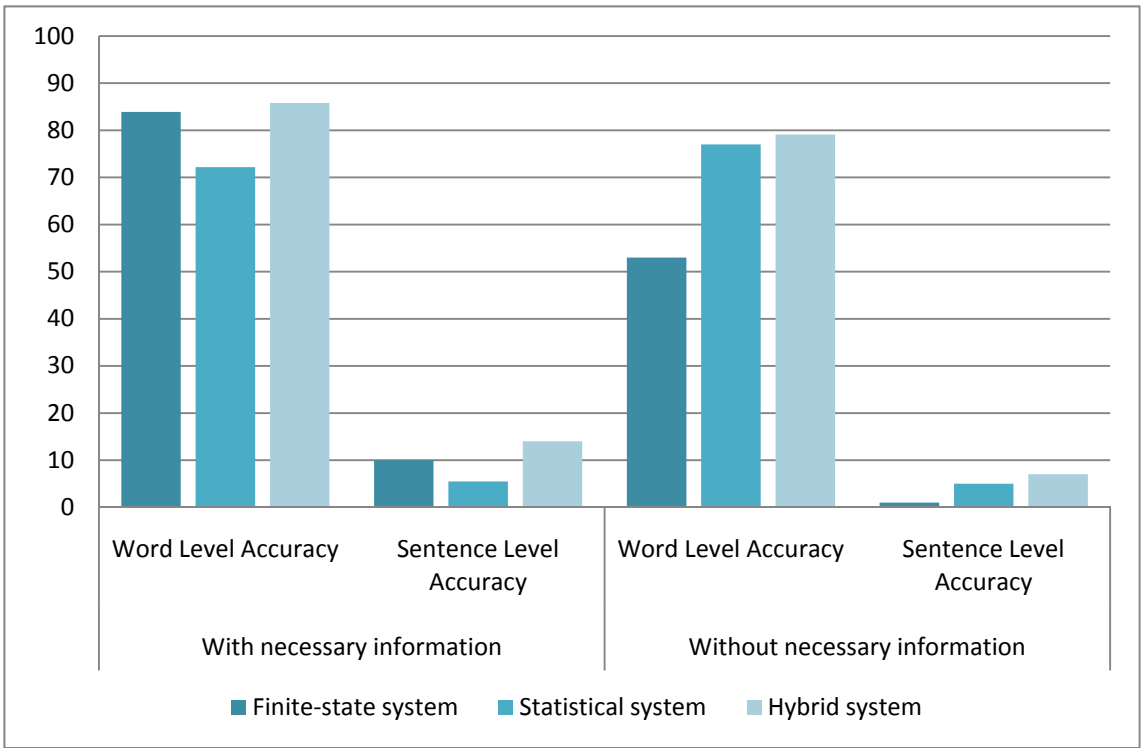


Figure 3 : Comparaison de systèmes de traduction scripturale d'ourdou en hindi aux niveaux des mots et des phrases

L'analyse de nos résultats de traduction scripturale montre que les évaluations subjectives, comme la satisfaction des utilisateurs, la facilité d'utilisation des résultats de la traduction dans la vie réelle, la fluidité et l'adéquation des textes traduits, sont également très importantes, en sus des évaluations objectives, comme la précision en mots en phrases, et la distance d'édition. Aux fins d'évaluation subjective, nous avons conçu des échelles différentes pour calculer les différentes mesures subjectives de nos systèmes de traduction scripturale.

Pour obtenir une estimation liée à la vie réelle pour nos systèmes de traduction, et améliorer l'efficacité de nos systèmes de traduction scripturale, nous avons également développé un système de traduction scripturale interactif en ligne. Ce système de traduction scripturale interactive sert à différents objectifs.

Bien que nous n'ayons pas encore été en mesure de développer des systèmes pratiques de traduction interdialectale, nous avons présenté une étude théorique sur la façon dont nous pouvons développer des systèmes de traduction interdialectale basés sur des approches de traduction automatique statistique (TAS) et de traduction syntaxique (basées sur le transfert syntaxique et sur un cadre linguistique). Cela comprend une étude sur le type de ressources linguistiques (lexiques et corpus parallèles) et de modules linguiciels (analyseurs morphologiques et syntaxiques) nécessaires pour la construction de systèmes de traduction interdialectale. La disponibilité de systèmes de traduction scripturale interactifs jouera un rôle essentiel dans le développement de ressources en ligne utilisant la très grande communauté Internet.

Nous avons essentiellement fait des expériences sur des langues Indo-Pak, qui représentent une grande part de la population du monde. La paire Ourdou-Hindi représente à elle seule 1.017 millions de locuteurs dans le monde entier. Seul le chinois compte plus de 1.000 millions de locuteurs. Le tableau 4 indique le nombre de locuteurs des langues Indo-Pak.

Sr.	Language	Number of Speakers
1	Hindi	853,000,000
2	Urdu	164,290,000
3	Punjabi	120,000,000
4	Sindhi	21,382,120
5	Seraiki	13,820,000
6	Kashmir	5,640,940
Total		**1178,133,060**

Tableau 4 : Nombre de locuteurs de langues Indo-Pak

Nous avons mis en ligne notre système de traduction scripturale pour hindi-ourdou, punjabi / shahmukhi-punjabi / gurmukhi, et seraiki/shahmukhi-seraiki/devanagari. Nous allons également mettre à disposition des systèmes de traduction scripturale pour sindhi/sindhi-sindhi/devanagari et cachemiri/ourdou cachemiri/devanagari dans un proche avenir.

Nous avons présenté des études théoriques pour le développement de la traduction interdialectale dans la troisième partie de cette thèse. À l'avenir, nous avons l'intention d'utiliser utiliser cette étude et de développer les ressources linguistiques et des modules linguiciels nécessaires pour le développement de systèmes de traduction statistique et syntaxique pour les langues Indo-Pak. Le Table 4 montre l'importance de cette étude en fonction de la taille des populations qui, nous l'espérons, pourront profiter de notre étude.

Résumé

Étant données une langue source L1 et une langue cible L2, un segment (phrase ou titre) S de n mots écrit en L1 peut avoir un nombre exponentiel $N=O(k^n)$ de traductions valides T1...TN. Nous nous intéressons au cas où N est très faible en raison de la proximité des formes écrites de L1 et L2. Notre domaine d'investigation est la classe des paires de combinaisons de langue et de système d'écriture (Li-Wi, Lj-Wj) telles qu'il peut y avoir une seule traduction valide, ou un très petit nombre de traductions valides, pour tout segment S de Li écrit en Wi. Le problème de la traduction d'une phrase hindi/ourdou écrite en ourdou vers une phrase équivalente en devanagari tombe dans cette classe. Nous appelons le problème de la traduction pour une telle paire un *problème faible de traduction*.

Nous avons conçu et expérimenté des méthodes de complexité croissante pour résoudre des instances de ce problème, depuis la transduction à états finis simple jusqu'à à la transformation de graphes de chaînes d'arbres syntaxiques partiels, avec ou sans l'inclusion de méthodes empiriques (essentiellement probabilistes). Cela conduit à l'identification de la *difficulté de traduction* d'une paire (Li-Wi, Lj-Wj) comme le degré de complexité des méthodes de traduction atteignant un objectif souhaité (par exemple, moins de 15% de taux d'erreur). Considérant la translittération ou la transcription comme un cas spécial de traduction, nous avons développé une méthode basée sur la définition d'une transcription intermédiaire universelle (UIT) pour des groupes donnés de couples Li-Wi, et avons utilisé UIT comme un pivot phonético-graphémique. Pour traiter la traduction interdialectale dans des langues à morphologie flexionnelle riche, nous proposons de faire une analyse de surface sur demande et limitée, produisant des arbres syntaxiques partiels, et de l'employer pour mettre à jour et propager des traits tels que le genre et le nombre, et pour traiter les phénomènes aux limites des mots.

A côté d'expériences à grande échelle, ce travail a conduit à la production de ressources linguistiques telles que des corpus parallèles et annotés, et à des systèmes opérationnels, tous disponibles gratuitement sur le Web. Ils comprennent des corpus monolingues, des lexiques, des analyseurs morphologiques avec un vocabulaire limité, des grammaires syntagmatiques du hindi, du punjabi et de l'ourdou, des services Web en ligne pour la translittération entre hindi et ourdou, punjabi (shahmukhi) et punjabi (gurmukhi), etc. Une perspective intéressante est d'appliquer nos techniques à des paires distantes LW, pour lesquelles elles pourraient produire efficacement des présentations d'apprentissage actif, sous la forme de sorties pidgin multiples.

Mots-clés:

Traduction Automatique, translittération automatique, problème faible de traduction, traitement multiscriptural, traitement multilingue, automates d'états finis, transducteurs d'états finis, méthodologie basée sur des règles, approche interlingue, transcription intermédiaire, approche basée sur les graphes, traduction interactive, morphologie, transformation morphologique, transformation mot-à-mot, analyse partielle en constituants, arbre syntaxique partiel, transformation d'arbres, méthodes empiriques, méthodes probabilistes, langues de l'Asie du sud, systèmes d'écriture, ourdu, hindi, punjabi, sindhi, cachemirien, seraiki.

Abstract

Given a source language L1 and a target language L2, a written translation unit S in L1 of n words may have an exponential number $N=O(k^n)$) number of valid translations T1…TN. We are interested in the case where N is very small because of the proximity of the written forms of L1 and L2. Our domain of investigation is the class of pairs of language and writing system combinations (Li-Wi, Lj-Wj) such that there may be only one or a very small number of valid translations for any given S of Li written in Wi. The problem of translating a Hindi/Urdu sentence written in Urdu into an equivalent one in Devanagari falls in this class. We call the problem of translation for such a pair a *weak translation problem*.

We have designed and experimented methods of increasing complexity for solving instances of this problem, from simple finite-state transduction to the transformation of charts of partial syntax trees, with or without the inclusion of empirical (mainly probabilistic) methods. That leads to the identification of the translation difficulty of a (Li-Wi, Lj-Wj) pair as the degree of complexity of the translation methods achieving a desired goal (such as less than 15% error rate). Considering transliteration or transcription as a special case of translation, we have developed a method based on the definition of a *universal intermediate transcription* (UIT) for given groups of Li-Wi couples and used UIT as a *phonetico-graphemic* pivot. For handling interdialectal translation into languages with rich flexional morphology, we propose to perform a limited on-demand surface analysis into partial syntax trees and to use it to update and propagate features such as gender and number and to handle word boundary phenomena.

Beside large-scale experiments, this work has led to the production of linguistic resources such as parallel and tagged corpora and of running systems, all freely available on the Web. They include monolingual corpora, lexicons, morphological analyzers with limited vocabulary, phrase structure grammars of Hindi, Punjabi and Urdu, online web-services for transliteration between Hindi & Urdu, Punjabi (Shahmukhi) & Punjabi (Gurmukhi), *etc*. An interesting perspective is to apply our techniques to distant L-W pairs, for which they could efficiently produce active learning presentations in the form of multiple pidgin outputs.

Keywords:

Machine Translation, Machine Transliteration, Weak Translation Problem, Multiscriptural processing, Multilingual processing, Finite-state Automata, Finite-state Transducers, Rule-based Methodology, Interlingua Approach, Intermediate Transcription, Graph-based Approach, Interactive Translation, Morphology, Morphological Transformation, Word-to-word Transformation, Partial Phrase Structure Analysis, Partial Syntax Tree, Tree Transformation, Empirical Methods, Probabilistic Methods, South Asian Languages, Writing Systems, Urdu, Hindi, Punjabi, Sindhi, Kashmiri, Seraiki.