Phonemic variability and confusability in pronunciation modeling for automatic speech recognition

Panagiota Karanasou

UNIVERSITÉ PARIS-SUD

ÉCOLE DOCTORALE D'INFORMATIQUE DE PARIS-SUD (ED 427)

Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur (LIMSI)

***DISCIPLINE** INFORMATIQUE*

THÈSE DE DOCTORAT

soutenue le 11/06/2013

par

Penny KARANASOU

Phonemic variability and confusability in pronunciation modeling for automatic speech recognition

Directeur de thèse :	Lori Lamel	LIMSI-CNRS
Co-directeur de thèse :	François Yvon	LIMSI-CNRS, Université Paris-Sud

Composition du jury :

Président du jury :	Anne Vilnat	LIMSI-CNRS, Université Paris-Sud
Rapporteurs :	Frédéric Béchet	Université Aix Marseille
	Eric Fosler-Lussier	Ohio State University
Examinateurs :	Lukas Burget	Brno University of Technology
	Denis Jouvet	INRIA Nancy - Grand-Est

Résumé

"Variabilité et confusabilité phonémique pour les modèles de prononciations au sein d'un système de reconnaissance automatique de la parole"

Penny Karanasou

Étant donné que la reconnaissance automatique de la parole (ASR) implique de plus en plus les utilisateurs finaux et élargit ses domaines d'applications, la recherche sur ASR évolue progressivement vers l'étude de la parole plus spontanée et moins canonique (par ex. le discours accentué). Cela crée un besoin croissant de mieux modéliser la variation de prononciation, qui est souvent insuffisamment traitée par le lexique de base. En outre, de nouveaux mots doivent être ajoutés dans le lexique, ce qui nécessite des moyens fiables pour générer automatiquement des prononciations pour ces mots hors-vocabulaire (OOVs). Dans la première partie de cette thèse, des approches inspirées de la traduction automatique statistique sont analysées pour générer automatiquement des prononciations pour des OOVs et pour enrichir le lexique de base avec des variantes de prononciation. Cependant, ajouter plusieurs prononciations par mot au vocabulaire peut introduire des homophones (ou quasi-homophones) et provoquer une augmentation de la confusabilité du système. La confusabilité de prononciations est plus élevée si toutes les prononciations sont équiprobables et peut potentiellement altérer la performance d'un système ASR. Une nouvelle mesure de cette confusabilité est proposée pour analyser et étudier sa relation avec la performance d'un système ASR. Jusqu'à ce point, des données audio ne sont pas prises en compte lors de la construction du dictionnaire de prononciation. La prochaine étape de cette thèse est de concevoir des techniques d'adaptation du dictionnaire permettant d'enrichir le lexique de base par rapport à un jeu de données particulier, tout en gardant la confusabilité minimum. Cette adaptation est traditionnellement appliquée à d'autres parties d'un système ASR (par ex. la modélisation acoustique). Un entraînement discriminant est utilisé pour entraîner les poids d'un modèle de confusion phonémique qui introduit des variantes de prononciation dans le système, contrebalançant ainsi le problème de confusabilité phonémique. La fonction objectif à optimiser est choisie afin de correspondre à la mesure de performance des différentes tâches étudiées: d'abord, une tâche ASR, puis, une tâche de détection de mots-clés (KWS). Pour la tâche ASR, une fonction objectif qui minimise le taux d'erreur au niveau des phonèmes est adoptée. Pour les expériences menées sur la détection de mots-clés, une approximation de la "ROC curve", directement liée à la performance de la détection de mots-clés, est optimisée.

Mots Clefs: modèles de prononciation, conversion graphème-phonème, confusabilité, apprentissage discriminant, reconnaissance automatique de la parole, détection des mots-clés

Abstract

"Phonemic variability and confusability in pronunciation modeling for automatic speech recognition"

Penny Karanasou

As Automatic Speech Recognition (ASR) makes its way to end users and widens its domains and ranges of applications, research in ASR is progressively shifting towards the study of more spontaneous and less canonical speech (for ex. accented speech). This creates an increasing need to better model pronunciation variation, which is often insufficiently covered by the baseline recognition lexicon. In addition, new word forms need to be entered in the lexicon, which requires reliable means to automatically generate pronunciations for these out-of-vocabulary words (OOVs). In the first part of this thesis, approaches inspired by statistical machine translation systems are investigated to automatically generate pronunciations for OOVs and to enrich the baseline lexicon with pronunciation variants. However, adding alternative pronunciations may introduce homophones (or close homophones), thereby increasing the confusability of the system. Pronunciation confusability is higher when all pronunciations are equally likely and can potentially harm the ASR performance. Next, a novel measure of this confusability is proposed and its relation with the ASR performance is investigated. Until this point, no speech data are taken into account during the construction of the pronunciation dictionary. The next step of this thesis work is to devise dictionary adaptation techniques, enabling to enrich an initial lexicon with respect to a particular dataset while at the same time keeping confusability low. Note that such adaptation is traditionally performed for other parts of an ASR system (i.e. acoustic modeling). Discriminative training is used to train the weights of a phoneme confusion model that introduces alternative pronunciations in the system, thus counterbalancing the phonemic confusability problem. The objective function to optimize is chosen to match the performance measure for the various tasks considered: first, an ASR task, then, a Keyword Spotting (KWS) task. For ASR, an objective that minimizes the phoneme error rate is adopted. For experiments conducted on KWS, an approximation of the ROC curve, directly related to the KWS performance, is maximized.

Keywords: pronunciation modeling, g2p conversion, confusability, discriminative training, speech recognition, keyword spotting

Acknowledgements

This thesis is for me the completion of four years of work but also of four enriching years of life. I feel lucky to have many people to thank for what they offered me during this period research-wise but also at a human level.

My first acknowledgements are due to my advisors, Lori Lamel and François Yvon. Lori Lamel first gave me the opportunity to work on the topic of pronunciation modeling, which was of great interest to me. She gave me a lot of freedom in exploiting diverse research directions and finding my way through, with a lot of trust in my choices which I appreciate. She also persuaded me of the importance of a strong experimental part of a research project and that working with real-world data can add value even to a theoretical-oriented work.

In a complementary way, François Yvon offered his valuable advice to my thesis in a moment when I needed more guidance and helped me find a direction to my work that was proven to correspond perfectly to my research interests. He is a great teacher, from whom I learned a lot on machine learning and on conducting research in a very organized and scientifically clear way. I am grateful to him for finding time to talk about any idea or question I might have had, explaining me the most complicated things in the most comprehensive way. But also for encouraging me when I needed that, always with a smile.

Next, I would like to thank Eric Fosler-Lussier and Frédéric Béchet for reviewing my thesis, as well as Anne Vilnat, Denis Jouvet and Lukas Burget for being members of my defense committee. Special thanks to E. Fosler-Lussier and D. Jouvet that provided detailed corrections and suggestions on my manuscript. To thank L. Burget, I start with the scientific knowledge he passed on to me while collaborating during my internship in the US, but also for his continuous interest in my work up until now making insightful comments and giving precious directions. He is the one that convinced me that in research there are no problems, but challenges. My gratitude to him goes a lot further though, since he became a friend listening to me and providing caring advice.

I would like to thank particularly Dimitra Vergyri who gave me the opportunity to spend some months in SRI International's STAR laboratory. Thanks to her I felt welcome from day one and I knew I had someone I could rely on being far from home. Meeting Dimitra and Lukas, people with great human qualities, and working with them was a great chance I had during this thesis. Thanks also to Murat Akbacak for introducing me to keyword spotting during the first weeks of my visit to the STAR laboratory and to Arindam Mandal for his help with the phoneme recognition experiments. I often think of everybody in this lab with gratitude for helping me integrate and feel like home.

Among the people that helped me in my work, I first refer to Jean-Luc Gauvain, the head of the Spoken Language Processing group of the LIMSI-CNRS laboratory, for his help on decoding and for welcoming me in the team. I am also indebted to Aurélien Max for introducing me to paraphases and the "pivot" method the first year of my thesis. And to Thomas Lavergne for our long discussions on CRF models and on programming in C++ (among other things related or not to research) and for a great collaboration when

working with him as a TA. But also for telling me to relax when I needed to do so, and for his convincing power on people to go for a drink after work (even if it did not always work with me).

In LIMSI I found an environment where doing research of high quality was combined with a lot of coffees, "pots" and laughters. First, I thank Thiago Fraga da Silva with whom I shared the same office from my first day in the lab. He was proven to be the best officemate I could have had, helping me, supporting me, sharing a wall with postcards with me and becoming a friend. I remember our discussions in the lab or in a bank in Paris. My thoughts go also to Nadi Tomeh for making people around him smile. I was always happy to pass by his office and find him there. Thanks also to Nadege Thorez for animating every discussion she participated in, and together with Ilya Oparin being able to talk seriously, but also joke, about almost every topic. I cannot forget to mention Josep M. Crego, with whom I felt soon at ease and could talk with in a trustful way. Artem Sokolov for his knowledges and curiosity to learn more in so many fields, and for helping me with C++ and OpenFst programming issues not leaving from my office until a solution was found. Hai Son Le for observing when I needed a boost of optimism and offering it to me. And Clement Chastagnol for our interesting discussions.

I also think of Alexander Allauzen for his "bon humeur", Guillaume Wisniewski for teaching me new greek cooking recipes, Hélène Maynard for her positive energy, Hervé Bredin for sharing some Voodoo donuts with me in Portland and Ioana Vasilescu for giving me some vitamins the last weeks of my thesis. Thanks also to Claude Barras for kindly responding to my questions on how a basic speech recognition system works the first months of my thesis. Other people I appreciate and would like to mention are Martine Adda-Garnier, Gilles Adda, Philippe Boula de Mareüil and Laurence Devillers.

From the new people that enriched our group the last year, I would like to mention Billy Hartmann who always had very interesting research ideas to share and who soon became the usual afternoon-break companion of Thiago and me. Thanks to Nicolas Pécheux for disassembling my laptop with me (and then reassembling it) and for being a warm person, passionate about his research and many other things. And to Benjamin Marie for announcing the lunch time. Finally to everyone else we shared good moments with, hoping I have not forgotten to mention them.

A part of this section must be dedicated to my friends outside the lab, including friendships that started in Greece and were preserved over the years, but also new friends I made along the way. From the people that have been there for quite a while, I thank Elli Katsarou for our frequent skype sessions and for being a caring person I can count on. Angeliki Koulouri and Alkistis Katsivali for our mutual visits all over Europe. Kostas Kaskavelis because our conversations have always been a source of inspiration. Giorgos Pierrakos for keeping still in touch despite the distance and for his valuable help with my resume. From my visit to the US, among some interesting people I met I would like to mention Gjergji Zyba, who took me under his protection as if I was his little sister. From the friends I made in Paris, I am grateful to Maro Kalantzopoulou for being there when I needed it the most and for bringing some family moments in my life in Paris. I would also like to thank Zoi Kaoudi, Asterios Katsifodimos and Despoina Trivela for discovering "bistronomie" together and for our long specialist-style discussions about everything

in life. And for spending together with Jesús Camacho-Rodríguez a lot of moments of carefree laughters. Yannis Katsis is added to the group for making the "gallette de roi" our personal new year's custom and for always calling me in our birthday. While Christina Zikou was always making me discover new places in Paris. Thanks also to Julien Hochart for reminding me the value of free time and free choices in life. And finally to Aurélien, who accompanied me in a big part of this thesis with all the love he could give.

I want to close my acknowledgements thanking my family, knowing that I cannot thank them enough. My brother, who is a solid presence in my life, whether in Paris, in Greece or further geographically. He has always known to give wise advice supporting and protecting me and always believing in me. My father, for making us his first priority in life, for his love and support and for reminding me that even after the darkest night there comes a new dawn. This thesis is dedicated to my mother, because I wouldn't be here if I had not seen in her eyes all I could become.

Table of Contents

Abstract iii

Acknowledgements vii

Table of Contents xi

List of Figures xv

List of Tables xvii

1 Introduction 1
- 1.1 Automatic Speech Recognition . . . 1
- 1.2 Pronunciation variation . . . 2
- 1.3 Grapheme-to-phoneme conversion . . . 5
- 1.4 The confusability problem . . . 6
- 1.5 Motivation . . . 6
- 1.6 Thesis outline . . . 7

2 Background and State-of-the-art 9
- 2.1 Grapheme-to-phoneme conversion . . . 9
- 2.2 Phonemic confusability . . . 12
  - 2.2.1 Confusability: an ASR error analysis . . . 13
  - 2.2.2 Moderating confusability . . . 14
  - 2.2.3 Speech-dependent lexicons . . . 15
  - 2.2.4 Combining g2p conversion and speech-dependent lexicons . . . 16
  - 2.2.5 Phonemic confusability in the Keyword-Spotting task . . . 18
- 2.3 FST background . . . 21
  - 2.3.1 Generalities . . . 21
  - 2.3.2 Semiring . . . 21
  - 2.3.3 Weighted Finite-State Transducers . . . 21
  - 2.3.4 Some useful semirings . . . 22
  - 2.3.5 Algorithms . . . 23
  - 2.3.6 Entropy semiring . . . 25
  - 2.3.7 Matchers . . . 26
  - 2.3.8 FST-based speech recognition . . . 26

3 SMT-inspired pronunciation generation **29**
3.1 Introduction . . . 29
3.2 Methodology . . . 30
3.2.1 Moses as g2p and p2p converter . . . 30
3.2.2 Pivot paraphrasing approach . . . 32
3.3 Experimental setup . . . 33
3.4 Evaluation . . . 35
3.4.1 Definition of evaluation measures . . . 35
3.4.2 G2P conversion results . . . 36
3.4.3 P2P conversion results . . . 40
3.5 Speech recognition experiments . . . 42
3.6 Conclusion . . . 45

4 Pronunciation confusability **47**
4.1 Introduction . . . 48
4.2 A new confusability measure . . . 49
4.2.1 ASR decoding with FSTs . . . 49
4.2.2 Decomposing the acoustic and linguistic modeling . . . 50
4.2.3 Definition of pronunciation entropy . . . 51
4.3 Phoneme recognition . . . 52
4.4 Pronunciation entropy results . . . 54
4.5 Conclusion . . . 56

5 Phoneme confusion model in ASR **59**
5.1 Introduction . . . 59
5.2 Problem set-up . . . 61
5.3 Training criteria . . . 62
5.3.1 The CRF model . . . 62
5.3.2 Soft-margin CRF . . . 63
5.3.3 Large-margin methods . . . 63
5.3.3.1 Perceptron . . . 63
5.3.3.2 Max-margin . . . 64
5.3.4 Optimization algorithm . . . 65
5.4 An FST-based implementation . . . 66
5.4.1 Preprocessing . . . 66
5.4.2 Defining the input and output FSTs . . . 67
5.4.3 Computing the edit distance with FSTs . . . 68
5.4.4 Discriminative training algorithms . . . 68
5.4.4.1 Perceptron . . . 69
5.4.4.2 Max-margin . . . 69
5.4.4.3 CRF . . . 69
5.4.4.4 Soft-margin CRF . . . 70
5.5 Experimental setup . . . 70
5.6 Phonemic analysis . . . 71
5.7 Evaluation . . . 72

5.7.1 Computation of the objective . . . 72
5.7.2 Phoneme Accuracy . . . 73
5.7.3 Decoding process . . . 74
5.7.4 Discussion of the results . . . 77
5.8 Conclusion . . . 78

6 Confusion model for KWS **81**
6.1 Introduction . . . 81
6.2 Keyword spotting system . . . 82
6.2.1 Indexing and searching representation . . . 82
6.2.2 Confusion model . . . 84
6.2.3 Confusion model initialization . . . 85
6.3 Confusion model training . . . 85
6.3.1 The Figure of Merit . . . 85
6.3.2 Discriminatively optimizing the Figure of Merit . . . 86
6.4 Experimental setup . . . 87
6.5 Results . . . 87
6.6 Conclusion . . . 89

7 Conclusion and Perspectives **91**
7.1 Thesis summary . . . 91
7.2 Perspectives . . . 92

Appendix A Phoneme set for American English **95**

Appendix B Publications **97**

Bibliography **99**

List of Figures

1.1 *Basic ASR system architecture* 3

2.1 *Basic KWS architecture* 18

3.1 *Alignment of the word "rainbow" with its pronunciation /renbo/* 31

4.1 *Example of ambiguity during word segmentation in ASR decoding* 49
4.2 *Expansion of the topology of the P FST with phi and rho matchers that consume the phonemes inserted between valid pronunciations* 51

5.1 *Perceptron loss on training data* 72
5.2 *CRF objective on training data* 73

6.1 *Timed Factored Index* 83
6.2 *ROC curves on training data* 88
6.3 *ROC curves on evaluation data* 88

List of Tables

2.1 *Basic matchers* 26

3.1 *Training conditions* 34
3.2 *PER on all references (canonical pron+variants) for Moses-g2p (M-g2p) and Pivot (P) for canonical pronunciation training* 37
3.3 *PER on all references (canonical pron+variants) for Moses-g2p (M-g2p) and Pivot (P) for the entire training set condition* 37
3.4 *Recall on all references (canonical pron+variants) and only on variants for Moses-g2p (M-g2p) and Pivot (P) for canonical pronunciation training* 38
3.5 *Recall on all references (canonical prons+variants) and only on variants for Moses-g2p (M-g2p) and Pivot (P) for the entire training set condition* 38
3.6 *Recall on variants only for generation of 1-, 4- and 9-best variants by Moses-g2p (M-g2p) and Pivot (P) for the entire training set condition* . . 39
3.7 *Macro-Recall on all references (canonical prons+variants) and only on variants for Moses-g2p (M-g2p) and Pivot (P) for both training conditions.* 39
3.8 *Results using Moses as phoneme-to-phoneme converter for the 3 training conditions* 40
3.9 *Results using the pivot paraphrasing method for the 3 training conditions* 41
3.10 *Quaero 2010 development data set composition and baseline word error rates (%) with the original dictionary and a single pronunciation one.* . . 43
3.11 *WER(%) adding Moses nbest-lists (M1, M2,M5) to the longest pronunciation baseline* 43
3.12 *WER(%) adding Moses nbest-lists (M1, M2,M5) to the most frequent pronunciation baseline* 44
3.13 *WER(%) adding Moses nbest-lists (M1, M2,M5) to the original LIMSI dictionary* 44
3.14 *WER(%) generating prons for OOVs using l2s and Moses nbest-lists (M1, M2, M5, M10).* 45

4.1 *Average number of states and arcs in the lattices and FSTs* 54
4.2 *Pronunciation entropy with different dictionary baselines* 54
4.3 *Pronunciation entropy with the 4-gram LM after adding n-best pronunciations, produced by a Moses-based g2p converter, to the "longest" baseline* 55
4.4 *Pronunciation entropy with the 4-gram LM after adding Moses' n-best pronunciations to the "most frequent" baseline* 55

4.5 *WER(%) adding Moses nbest-lists (M1, M2,M5) to the single pronunciation dictionaries* 56

5.1 *Phoneme Accuracy in the dev set* 73
5.2 *Phoneme Accuracy in the test set* 73
5.3 *Phoneme Accuracy of the phoneme output of the word recognizer (test set)* 75
5.4 *Word Accuracy on the test set* 77

A.1 *Phoneme set for American English* 95

Chapter 1

Introduction

This thesis starts with an introduction to the problem of pronunciation modeling for automatic speech recognition (ASR). As the area of speech processing applications expands, pronunciation modeling, together with speech recognition, are characterized by a progressive shift of the processed data types from read speech to spontaneous talk, and of the used methods from knowledge-based to principally statistical ones. Despite a long history of research in the field, construct a good pronunciation dictionary is still an open problem. Many decisions are required in the process of building a dictionary. Examples of such decisions, to name a few, include the size of the dictionary, the sources of variation to take into account, the number of variants to include, the use or not of information from other sources such as prosodic or syntactic knowledge, a suitable way to handle the homophones, etc. Consequently, there is still a vivid interest in improving the construction of the recognition lexicon, and there are still improvements to be made, especially when spontaneous or less canonical (i.e., accented) speech is involved.

1.1 Automatic Speech Recognition

Automatic speech recognition is the task of transcribing a recorded speech segment into its corresponding sequence of words. A general description of an ASR system will be given following (Young, 1996). The overall procedure realized in an ASR system is presented in Figure 1.1, taken from (Holmes and Holmes, 2002). A front-end signal processor is used to convert a speech waveform into a sequence of acoustic vectors, $Y = y_1, y_2, ..., y_T$. Each of these vectors is a compact representation of the short-time speech spectrum covering a period of typically 10 msecs. The utterance consists of a sequence of words $W = \mathtt{w}_1, \mathtt{w}_2, ...\mathtt{w}_\mathtt{T}$ and it is the job of the ASR system to determine the most probable word sequence $\hat{W}$ given the observed acoustic signal Y. To do this, Bayes' rule is used to decompose the required probability $P(W|Y)$ into three components, that is:

$$\hat{W} = \arg\max_W P(W|Y) = \arg\max_W \frac{P(W)P(Y|W)}{P(Y)}. \tag{1.1}$$

And if we want to approximate the decision rule to exploit also alternative pronunci-

ations, it becomes:

$$\begin{aligned} \hat{W} &\cong \arg\max_{W} \sum_{q} P(W)P(S_q|W)P(Y|S_q) \\ &\cong \arg\max_{W} \max_{q} P(W)P(S_q|W)P(Y|S_q), \end{aligned} \tag{1.2}$$

where $q = 1, 2, ..., N$ is the index to multiple phoneme sequences S_q that are alternative pronunciations for sentence W.

The term $P(W)$ represents the a priori probability of observing the word sequence W independent of the speech signal and is determined by the language model. The term $P(Y|S_q)$ gives the probability of observing the vector sequence Y given a phoneme sequence S_q and is described by the acoustic model. Finally the term $P(S_q|W)$ gives the probability of observing the phoneme sequence S_q given the word sequence W and is described by the pronunciation model.

Finding the best word sequence $\hat{W}$ is the task of the decoder. There are two main approaches: depth-first and breadth-first. In depth-first designs, the most promising hypothesis is pursued until the end of the speech utterance is reached. Some examples of depth-first decoders are stack decoders and A*-decoders (Jelinek, 1969),(Paul, 1991). In breadth-first designs, all hypotheses are pursued in parallel. Breadth-first decoding exploits Bellman's optimality principle and is often referred to as Viterbi decoding (Aubert, 2000). Large-vocabulary ASR systems are complex and pruning of the search space is essential; this typically uses a process called beam search (Steinbiss, 1989).

The ideal decoding strategy would use every available knowledge source. But is often difficult or expensive to integrate a very complex knowledge source into first pass search. Hence, multi-pass decoders are used (Gales and Young, 2007). The output of the first recognition pass is generally expressed as a rank-ordered n-best list of possible word sequences, or as a word graph or a lattice describing all the possibilities as a network. A multi-pass decoding is inappropriate for real-time applications but it has shown significant improvements for off-line applications.

1.2 Pronunciation variation

Pronunciation variations can be modeled at different levels of ASR systems described above, as explained in (Strik and Cucchiarini, 1999): the lexicon, the acoustic model, the language model. At the acoustic level, context-dependent phone modeling captures the phone variations within particular contexts. At the lexicon level, alternative pronunciations can be included to explicitly represent pronunciation variation. At the language model (LM) level, the inter-word pronunciation variations are handled with grammar networks, statistical LMs or multi-word models.

The growing interest in automatic transcription of Conversational Telephone Speech (CTS) has stimulated research in modeling pronunciation variation. It has been found that pronunciation variation is one of the factors that influences the performance of an automatic speech recognition system (McAllaster et al., 1998), especially for spontaneous speech. Indeed, there is a large number of possible pronunciation variants occurring in

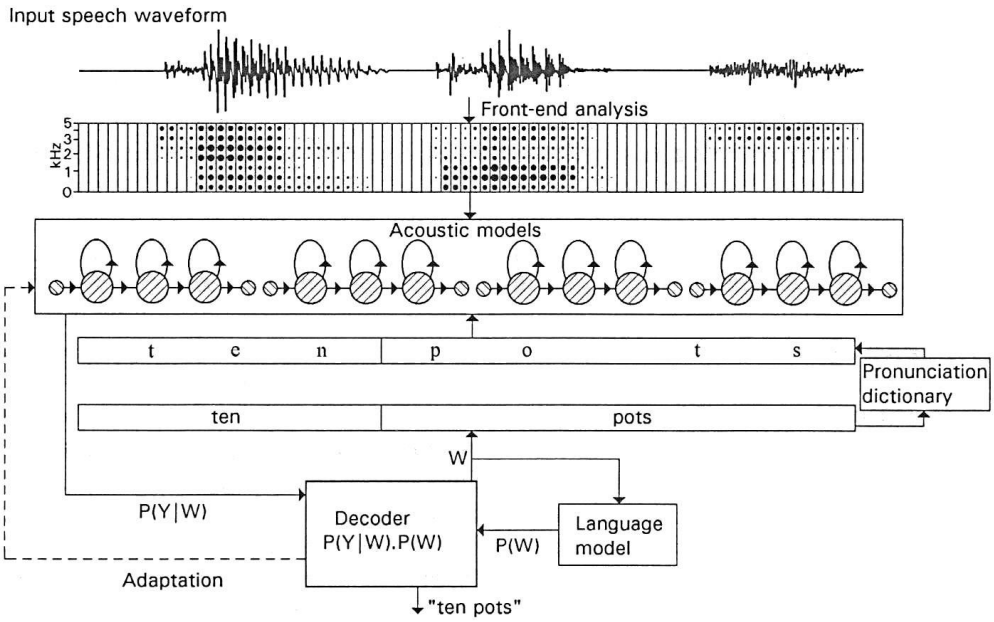

Figure 1.1: *Basic ASR system architecture*

spontaneous speech; these variants often extend beyond single speech sounds (modeled by the acoustic model) and reach up to whole words or word tuples. Even context-dependent acoustic models for sub-word units (like phonemes) are not able to cover pronunciation variants of this kind (Kipp et al., 1997). Some examples of phonetic variability not captured by (or presenting difficulties for) triphones (context-dependent phones) include syllable deletion, vowel reduction (prosodic effects) and other cases (non-reduction) of phone substitution (mostly because of co-articulation). Thus, pronunciation variation is usually modeled by enumerating appropriate pronunciations for each word in the vocabulary using a pronunciation lexicon.

Following (Ladefoged, 2006), two levels of representation of the pronunciation variation are distinguished: the broad and the narrow phonetics. The broad phonetics cover only the more noticeable phonetic features of an utterance, whereas narrow phonetics encode more information about the phonetic variations of the specific allophones in the utterance. One particular form of broad phonetics is a phonemic transcription, which disregards all allophonic difference. This is the kind of transcription that is used in the recognition dictionary we will be working with in this thesis (Lamel and Adda, 1996). The narrow phonetics are modeled by the allophones of the acoustic model and are not the subject of research in this work. It should be noted, however, that there is a continuum between broad and narrow phonetics, and there are cases where it is difficult to do the distinction between them.

Examples of broad phonetics are variations that depend on word-level features of lexical items (such as part-of-speech, case, tense, etc.) and variations that are particular to specific lexical entries. We give some examples of these variants specifically for English, which is the language used for the experimental part of this thesis. Variants that depend

on word-level features include contractions (“what’s”, “can’t”, etc.), reductions (“gonna”, “wanna”, etc.), part-of-speech variants (as in the noun and verb versions of “record”), and tense variants (as in the past and present tense versions of “read”). In most speech recognition systems, these types of variants are handled in an explicit manner. Reductions and contractions are typically entered into the pronunciation lexicon as distinct entries independent of the entries of their constituent words. All alternate pronunciations due to part of speech or tense are typically entered into the pronunciation lexicon within a single entry without regard to their underlying syntactic properties.

Other variants modeled by broad phoentics are simply word-dependent pronunciation variants which are not the result of any linguistic features of that word. A simple example of a word with such variants is “either”, which has two different phonemic pronunciations. These variants are typically encoded manually by lexicographers.

Narrow phonetics include variations that depend on the stress and syllable position of phonemes In English these are typically related to the realization of stop (or plosive) consonants. The set of possible allophones of a stop consonant in English is heavily dependent on its position within a syllable and the stress associated with the syllables preceding and following the stop. For instance, the stop /t/ has several allophonic realizations. It is an aspirated phone [t^h] when word-initial (“tea”) and in front of stressed syllables. It is glottalized when syllable-final or before nasals [t'] (“eat”). In American English, intervocalically when the second vowel is unstressed, the /t/ sounds a lot softer an is pronounced as a flap [ɾ] [1](“butter”).

Another case of narrow phonetics are variations that depend only on local phonemic or phonetic context (for ex., variants of “coupon”). These are independent of any higher-level knowledge of lexical features, lexical stress, or syllabification. Examples of these effects are vowel fronting, place assimilation of stops and fricatives, gemination of nasals and fricatives, and the insertion of epenthetic silences.

It is important to note that pronunciation variation can result from several sources, such as human error (i.e., mispronunciations), the linguistic origin of the speaker (regional variation or foreign accent), the speaker’s education and socio-economic level, the speaking style and conversational context and the relationship between interlocutors (Coetzee and Kawahara, 2012). We do not explicitly account for these types of variations within the grapheme-to-phoneme conversion framework presented in Chapter 3 of this thesis. However, the methods presented in Chapters 5 and 6 could be used to adapt a lexicon to a particular accent or speaker if suitable data were available.

Predicting pronunciation variations, that is, alternative pronunciations observed for a linguistically identical word, is a complicated problem and depends on a number of factors that will be detailed in the following paragraphs. The construction of a good pronunciation dictionary is critical to ensure acceptable ASR performance (Lamel and Adda, 1996). Moreover, the number of pronunciation variants that need to be included in a dictionary depends on the system configuration. For example, (Adda-Decker and Lamel, 1999) noticed that, for both read and spontaneous native speech recognition, better acoustic models seem to demand less pronunciation variants incorporated in the lexicon,

1. A flap is a type of consonantal sound, which is produced with a single contraction of the muscles so that one articulator (such as the tongue) is thrown against another.

because acoustic models can model implicitly a part of the phonemic variation. They also noted that consistency is one of the most important factors of pronunciation modeling.

What makes pronunciation modeling even more complicated, especially for conversational speech, is the fact that often there is only a partial change in phones which cannot be well represented neither by baseforms nor by surface-forms. In (Saraclar and Khudanpur, 2004) it is noted that further analysis of the manual phonetic transcription reveals a significant number ($> 20\%$) of instances where even human labelers disagree on the identity of the surface-form, as the sound of alternative pronunciations often lie somewhere between the canonical and alternative pronunciation. The best strategy to adopt to model pronunciation variation therefore depends on the degree of deviation from the canonical pronunciation.

Another interesting observation that makes pronunciation modeling a non-trivial problem is that variation depends also on the frequency of words. (Antilla, 2002) claims that an adequate theory of phonology should account for the "locus of variation" (where variation is observed and where it is not), and the "degrees of variation" (the frequency of different variants). In other words, we should not add the same number of variants to all words, but allow some variation phenomena more to certain words. For example, frequent words seem to have more variants in spontaneous speech. This is because speakers seem to be less careful in pronouncing frequent words and words that are very easily predictable based on the context (Fosler-Lussier, 1999).

1.3 Grapheme-to-phoneme conversion

Grapheme-to-phoneme conversion (g2p) is the task of finding the pronunciation of a word given its written form. Despite several decades of research, it remains a challenging task and plays a role in many applications for human language technologies. It is an essential element of speech recognition and speech synthesis systems. Moreover with the wide use of real data there are words not yet included in a recognition dictionary (out-of-vocabulary words), for which a pronunciation rapidly and automatically generated is often required. These words are often named entities which depend strongly on current events and, thus, it is difficult to predict their presence in speech and include them in the dictionary (Béchet and Yvon, 2000). Another domain of application of the phonetization task in natural language processing is the detection and correction of orthographic errors (van Berkel and De Smedt, 1988). What is more, the strong relation between phonology and morphology is well known and studied with morphological phenomena of purely phonological origins or guided by phonological constraints, among other dependencies (Kaisse, 2005). Other applications include computer-aided language learning, pronunciation training and in general e-learning systems.

Grapheme-to-phoneme conversion is a way to generate pronunciations totally independent of a given set of speech data. The pronunciation model is often the only part of an ASR system that is static and not trained on a particular data set. However, the need of a dynamic dictionary adapted to speech data is found to be important to improve ASR performance. Adding a lot of variants to a recognition dictionary, even if these variants are evaluated to be of high quality, does not necessarily lead to an improvement in the

error rate of the system. The alternative pronunciations introduce a confusability to the system that is difficult to measure and to limit in an appropriate way. Adding alternative pronunciations with probabilities can moderate this confusability, but training these pronunciation weights is an open problem.

1.4 The confusability problem

When adding alternative pronunciations to a lexicon there is always the potential of introducing a detrimental amount of confusability. The homophone (words that sound the same but are written differently) rate may increase, which means that these additional variants may not be helpful to the recognition performance (Tsai et al., 2001). A typical example in English is the word "you" (see Appendix A for the phoneme notations): the received pronunciation is */yu/* and is chosen when one single variant is used; modeling some variation requires to consider the pronunciations */yu/* and */yc/*, which both occur in our multiple pronunciation dictionary. The latter pronunciation (*/yc/*), in the phrase "you are", is easily confused with */ycr/*, the pronunciation of "your". Such confusions, in particular when they involve frequent words, can cause a degradation of the ASR system as more alternatives are added.

To handle this confusability, the phonemic variation provided by speech data can be useful in the process of choosing which variants to add in the recognition lexicon and of training their weights. In this way, an adapted dynamic lexicon is constructed. This lexicon is ideally trained to minimize the error rate of the system, as is traditionally done for the training of the acoustic model and also tried for the training of the language model.

1.5 Motivation

Despite the abundance of work on the automatic generation of pronunciations and pronunciation variants, there is still place for improvement. Many techniques proposed to handle the generation of pronunciations for out-of-vocabulary words (OOVs) do not consider how to model pronunciation variation. The first part of this thesis jointly addresses the prediction of baseline and variant pronunciations for OOV words, as well as the enrichment of the dictionary with pronunciation variants for the existing words, by borrowing techniques from statistical machine translation-in effect, modeling pronunciation variation as we would a foreign language. These variants are often necessary, especially when working with conversational speech where reduced forms of pronunciations frequently replace the canonical forms. In this part of the thesis, a given dictionary is used as the training corpus and no speech data are involved in the training procedure.

A recognition lexicon introduces a certain amount of ambiguity to the decoding procedure because of homophones that can be present at the word level, but often expand up to the sentence level. This ambiguity augments when variants are added to the lexicon. Subsequently, the focus of this thesis moves towards analyzing this confusability introduced to the system by a pronunciation dictionary containing or not pronunciation variants. A measure is proposed to model this confusability at the sentence level using

posterior estimates, and its correlation with the error rate of the ASR system is investigated.

Trying to find a way to better moderate this confusability, next speech data are also taken into account for the construction of the recognition lexicon, which this time is adapted to a particular audio data set, as is traditionally done for the acoustic modeling. In this case, a discriminative framework aiming to directly minimize the error rate of the system is chosen in order to decide which variants to keep in the lexicon and how to suitably train their weights.

Finally, a part of this work is dedicated to the keyword spotting (KWS) task which has gained a lot of attention the last years as the amount of available audio data augments and an efficient way to index and search them becomes crucial in order to have access to them. This is a post-processing task of speech recognition, using the output of an ASR system to construct an index on which a user-defined query term is searched in order to detect if and when it was uttered. A discriminative framework is adopted again in order to expand the index of a KWS system adding phonemic variation to it and correcting ASR errors.

1.6 Thesis outline

Below we present an overview of how this thesis is organized, as well as the main contributions of each chapter. This document is organized in six chapters, each devoted to a particular aspect of this work.

Chapter 2 provides the necessary background to follow the rest of the thesis. Besides, it discusses related work on grapheme-to-phoneme conversion, phonemic confusability and speech-dependent pronunciations as well as phonemic confusability as it appears in the keyword spotting task.

Chapter 3 will focus on the automatic generation of pronunciations for OOVs, and of pronunciation variants when the baseform is given. In this chapter, the main contribution is two proposed methods that give state-of-the-art results for the grapheme-to-phoneme and the phoneme-to-phoneme tasks.

Chapter 4 deals with understanding more on the problem of confusability introduced when a multiple pronunciation dictionary is used. A new measure is proposed to quantify this confusability and its correlation with the ASR performance is discussed.

Chapter 5 proposes a discriminative framework for expanding the pronunciation dictionary using speech data. A confusion model of phonemic pairs is used to expand the phonemic space where pronunciations are looked for during ASR decoding. The weights of this confusion model are discriminatively trained to minimize the phoneme error rate.

Chapter 6 expands the use of a discriminatively trained phonemic confusion model to the keyword spotting task. This time the objective function to minimize is an approximation of Figure of Merit (FOM), a measure of the keyword spotting performance.

Chapter 7 gives a final summary of this thesis and some reflections about the presented work, as well as some future research directions.

Chapter 2

Background and State-of-the-art

This chapter provides an overview of the methods used for automatic generation of pronunciations and pronunciation variants. A variety of methods have been proposed that can be broadly grouped into knowledge-based and data-based methods. We can further distinguish two families of approaches in data-driven pronunciation generation. The first one represents the case when a pronunciation dictionary (usually manually generated using linguistic knowledge) is used as the training set to generate pronunciation variants and new pronunciations for out-of-vocabulary (OOV) words. These tasks are usually referred to in the literature respectively as grapheme-to-phoneme (g2p) conversion and phoneme-to-phoneme (p2p) conversion (when only variants of existing baseform pronunciations are concerned)(Section 2.1).

However, adding pronunciation variants to an ASR system does not always improve the performance, independently of how these variants are produced. The reason is that confusability is introduced to the system by the alternative pronunciations. And while all the other parts of an ASR system are trained to be adapted to particular data and minimize the system error, this is not the case for the components and the weights of the pronunciation model. That is why there have been some attempts to generate pronunciations dependent on those observed in speech data and suitably train their weights. Different machine learning methods have been explored in this direction. Lately discriminative training has been used and appears to better counterbalance the introduced confusability. This second family of speech-dependent approaches will be detailed in Section 2.2. There will be a presentation of the existing literature for the ASR task and also the post-processing ASR task of keyword spotting (KWS) which is also explored in this thesis.

An important part of the work presented in this thesis on measuring pronunciation confusability and on the discriminative training of pronunciation weights is implemented in a Finite State Transducer (FST) framework. Thus, some basic elements of the theory of FSTs are presented in Section 2.3 of this chapter.

2.1 Grapheme-to-phoneme conversion

Finding the pronunciation of a word from its written form (g2p conversion) has long been a research topic and has many applications, especially in speech synthesis and recog-

nition, and spelling checkers. For many applications it is important to predict pronunciations of new terms or to add alternative pronunciations for existing ones. One of the problems we are trying to solve in this way is the problem of OOVs. In speech recognition, OOVs are never recognized and are substituted by in-vocabulary words. This can lead to misrecognition also of neighboring words, and thus to errors that cannot be recovered by later processing stages. Concerning the need for alternative pronunciations, it is more crucial in conversational speech when often reduced forms of pronunciations are used. This, of course, is highly related to the articulatory characteristics and to the linguistic background of a particular speaker.

Historically a substantial amount of manual effort is involved in the creation of a pronunciation lexicon. Knowledge-based methods using phonological rules (Oshika et al., 1975), (Divay and Vitale, 1997), (Adda-Decker and Lamel, 1999), (Wester, 2003), require specific linguistic skills, are not language-independent and do not always capture the irregularities in natural languages, especially in spontaneous speech. These phonological rules may be implemented implicitly, in that the various alternative pronunciations that they describe are specifically listed as entries in the lexicon. Or they can be implemented explicitly in the system. That is, apply transformation rules on the baseform in order to create alternative pronunciations. Another drawback of these methods is that, there may be a mismatch between the information found in the linguistic literature and the data for which the method has to be used. For example, the linguistic resources may not cover variation found in spontaneous speech (Strik and Cucchiarini, 1999).

However, with the large vocabularies used in automatic systems, there has been a move towards data-driven approaches, based on the idea that given enough examples it should be possible to predict the pronunciation of an unseen word simply by analogy. In this section we will present some methods for automatically constructing a static pronunciation dictionary. Meaning that the only training data available are the pairs of words and their phonemic transcriptions in a dictionary. No use of speech data will be made to learn from real spoken utterances. This last case will emerge as a need because of the drawbacks of static dictionaries and will be examined in details in Section 2.2.

A variety of machine learning techniques have been applied to the g2p problem in the past including neural networks (Sejnowski and Rosenberg, 1986) and decision trees (Dietterich and Bakiri, 1995) that predict a phoneme for each input letter using the letter and its context as features, but do not consider -or consider very limited- context in the output. More complex configurations of decision trees determine the questioned grapheme context in a dynamic way as the tree grows (Pagel et al., 1998). Another type of tree is the classification and regression tree (CART), thoroughly studied by (Breiman et al., 1984). This kind of tree is grown by testing a set of binary questions for each node, and choosing the best question according to some measures. Generalized decision trees (Vazirnezhad et al., 2005), simultaneously take into account word phonological structures, stress, and phone context information and typically achieve better results.

Other techniques allow previously predicted phonemes to inform future decisions such as HMM in (Taylor, 2005) but they do not take into account the input's graphemic context. With this model, phonemes are regarded as states that form a Markov chain, from which grapheme observations are drawn independently. The task of pronunciation prediction is

then to find the optimal state sequence within the HMM, given a spelling of a word as the observation sequence. All the above mentioned methods tend to degrade rapidly when encountering patterns unusual for the language under consideration.

Another proposed method is the so-called "pronunciation by analogy" method (Dedina and Nusbaum, 1991), (Yvon, 1996), based on formal analogies in the graphemic domain, allowing derivation of the correct pronunciation for a new word from the parts of similar words present in the dictionary. Given a suitable measure of similarity between words, such as the Levenshtein distance, they directly retrieve partial pronunciations for local fragments of the input word. These pronunciation fragments are then concatenated to obtain the final pronunciation. The pronunciation by analogy allows a better handling of unusual patterns, but relies heavily on individual, language-dependent alignments between letters and phonemes (Bellegarda, 2005).

A g2p converter considered in the literature to give state-of-the-art results is one based on joint-sequence models, originally proposed by (Deligne et al., 1995) and applied to g2p conversion by various researchers such as (Bisani and Ney, 2002) and (Chen, 2003). Such converters achieve better performance by pairing letter sub-strings with phoneme sub-strings, allowing context to be captured implicitly by these groupings. Other methods using many-to-many correspondences, such as the one proposed in (Jiampojamarn et al., 2008) report high accuracy.

Following the reasoning of (Wang and King, 2011), we can observe that of the models described above, HMMs and joint-sequence models are generative models, i.e., they model the joint probabilities of graphemes and phonemes and derive the posterior probability according to the Bayes' rule. On the other hand, neural networks and decision trees are discriminative models which estimate the posterior probability directly. From another perspective, HMMs and joint-sequence models perform global inference, meaning that they search for the optimal pronunciation as an entire phoneme sequence (even if this is based only on a sliding localized window of grapheme observations), while neural networks and decision trees perform piece-wise inference to generate pronunciations for individual graphemes and then compose the word pronunciation by concatenation. A discriminative model may be superior to a generative model because it does not need to model the (possibly complex) distributions over observations; on the other hand, global inference will probably be superior to the piece-wise inference. The CRF model has been used successfully in speech recognition lately. The idea behind using it also for g2p conversion is that it is a discriminative model that can perform global inference, suggesting that it may be more suitable for g2p than decision trees, HMMs or joint-sequence models. Some examples can be found in (Lavergne et al., 2010), (Wang and King, 2011), (Illina et al., 2011) and (Lehnen et al., 2011) reporting state-of-the-art performance.

Another interesting approach is the one presented in (Chen et al., 2008). They utilize discriminative training, which has been successfully used in speech recognition, to sharpen the baseline n-grams of grapheme-phoneme pairs generated by a standard joint-sequence model. They address the problem in a unified discriminative framework. Two criteria, maximum mutual information (MMI) and minimum phoneme error (MPE), are investigated and improvements are shown on the test datasets.

Recently, g2p conversion has also been expressed as a statistical machine translation

(SMT) problem. Moses, a publicly available phrase-based SMT toolkit (Koehn et al., 2007), was used for g2p conversion and tested in French (Laurent et al., 2009) and Italian (Gerosa and Federico, 2009) ASR systems and other languages (Rama et al., 2009). This method is based on the analogy of the problem of g2p conversion and SMT. It uses many-to-many alignments between the graphemic and the phonemic transcriptions and, thus, allows to take into account context information in both sides. SMT-based methods also give state-of-the-art results in g2p conversion.

Concerning the p2p conversion, the same methods used for g2p conversion can be applied. Data-based methods proposed in the literature include the use of neural networks (Fukada et al., 1999), decision trees (Weintraub et al., 1996), and also automatic generation of rules (van den Heuvel et al., 2009) or confusion matrices (Tsai et al., 2007) for p2p conversion. Sometimes, specific-purpose p2p converters can be used to correct mistakes of a general-purpose g2p, for example to improve the pronunciation of proper names (van den Heuvel et al., 2007).

All the above mentioned methods predict individual phonemes or subsequences of phonemes. Rather than predicting the full phonemization of a word given its orthography in one go, they decompose or simplify the task. This task simplification mitigates sparseness problems of the machine learning systems. In addition, task decomposition allows them to be robust and generic when they process unseen data. However, these systems tend to perform badly when there are many low-frequency and too case-specific classes. Another important drawback is the lack of a global method to check whether their local decisions form a coherent output. The authors of (van den Bosch and Canisius, 2006) give the examples of a g2p conversion system that predicts schwas on every vowel in a polysyllabic word such as “parameter” because it is uncertain about the ambiguous mapping of each of the “a”s and “e”s, and thus produces a bad pronunciation. Global models that coordinate, mediate, or enforce that the output is a valid sequence are typically formulated in the form of linguistic rules, applied during processing or in post-processing, that constrain the space of possible output sequences (see for example (Prince and Smolensky, 2004)). In (van den Bosch and Canisius, 2006), a method is proposed to learn constraints automatically from data in a global model.

To sum up, the need to model the g2p conversion in a way that takes both graphemic and phonemic contexts into account is observed in order to obtain state-of-the-art results. Lately, there is also a turn towards methods that perform global inference and are based on discriminative modelization and high accuracy is achieved. An open issue is still how to control the validity of the generated pronunciations, maybe integrating global word features to the explored models.

2.2 Phonemic confusability

The problem of simply adding pronunciation variants generated by a g2p or a p2p converter to a lexicon without imposing any constraints, is the potential increase of confusability introduced to the system. In this section, we will see that this confusability is an inevitable phenomenon of natural speech. We will then present methods proposed to analyze this confusability. In this direction, an analysis of the ASR errors related to con-

fusions will be attempted. We will continue with an overview of the literature concerning possible ways to constrain the detrimental influence of confusability to ASR performance. There will be a focus on attempts to generate dynamic lexicons based also, or completely, on speech data. This section concludes with a detailed presentation of the phonemic confusability problem in the keyword spotting task.

2.2.1 Confusability: an ASR error analysis

Work on pronunciation modeling for automatic speech recognition systems has had mixed results in the past; one likely reason for poor performance is the increased confusability in the lexicon that results from just adding new pronunciation variants. In addition, increasing the number of pronunciations within a system often increases the decoding time. Both of these problems are related to the concept of confusability: words with similar phonetic sequences can be confused with each other. With more pronunciations, the homophone rate (words that sound the same but are written differently) increases, which means that these additional variants may not be helpful to the recognition performance (Tsai et al., 2001). Such cases, in particular for frequent words, can be responsible for the degradation of the ASR system when many alternatives are added.

At this stage, it could be interesting to see how the confusions are perceived by humans and if there is a correspondence between these human methods of perception and the way confusion is perceived and analyzed in an ASR system. The problem of identifying the unit of human acoustic perception is still an open issue, but there have been a lot of studies that describe the confusable segments for the human auditory system taking a sub-word (phonetic unit) approach. In segmental phonology, phonemes are the sounds which can distinguish one word from another. A phoneme is defined as a contrastive phonological segment (Chomsky and Halle, 1968). However, most machine-based perception methods perform the confusability analysis at the word level, limiting themselves to observed spoken examples (Mangu et al., 2000), (Goel et al., 2004). One attempt to address this weakness is the thorough descriptive work of (Greenberg et al., 2000), in which the outputs from eight recognition systems are compared at the phonetic level. Their analysis shows that phonetic and word errors are correlated and conclude that acoustic-phonetic variation is a leading contributor to word errors. The missing link in this work is, however, an analysis of how the phonetic variability affects word recognition. A second drawback of these methods is that they are based on an a posteriori analysis of the speech recognition errors and are unable to make any predictions (Printz and Olsen, 2000), (Deng et al., 2003). For example, (Chase, 1997) developed a technique for assigning blame for a speech recognition error to either the acoustic or language model. This error assignment allows system developers to focus on certain speech segments for improving either of the two models; however, one cannot use this model to generate new predictions. Thus, the capability to generalize to unseen speech data is missing and restrains the use of such techniques.

For the reasons exposed in the above paragraph, we will focus on the analysis algorithms that treat confusion data at a sub-word level. But now we will try to circumvent the problem of having a descriptive model of ASR errors which cannot be generalized

to speech data not yet observed. One option is to compute statistics at the acoustic level as done in (Printz and Olsen, 2000). However, in this work the only source of confusion considered is the overlap between Gaussians, not taking into accounts cases where the acoustics might change dramatically (i.e., presence of noise). In addition, this method requires a very detailed knowledge of the system, a fact that makes the generalization of the use of such a measure a difficult task. An alternative is to collect statistics about pairs of recognized and correct linguistic units (e.g., phonemes) and to estimate the confusions at this level rather than at a more detailed level. One might learn that the recognizer often substitutes one phone for another, or often deletes a particular phone, resulting in a recognition error. This is the approach taken in (Deng et al., 2003) in estimating the word error rate for a particular task, and is similar to the approach taken by (Fosler-Lussier et al., 2002). While this type of modeling produces a less detailed analysis than comparing the distance between mixtures of Gaussians, it requires fewer assumptions about the recognizer (for example, it can be used to model systems with neural network acoustic models, or more complex structures such as general graphical models (Bilmes and Zweig, 2002)). This kind of models of confusion should be easier to train (and thus may be more usable by non-experts), but the accuracy of the predictions may suffer. For example, in (Pucher et al., 2007), it is shown that the confusion measure that presents the higher correlation with the word confusion is an HMM-based distance. However, edit distance between phonemes performs quite well especially when articulatory and perceptual phonetic information is used in addition. Finally, in (Jyothi and Fosler-Lussier, 2009) better error predictability is achieved by combining HMM-based phone distances with information providing by phone confusion matrices.

2.2.2 Moderating confusability

Treating the confusability at the pronunciation level has proven to be quite a complicated task and a straight correlation with the word error rate of the ASR system is yet to be discovered. A straightforward solution could be to discard any homophones. For example, in (Sloboda and Waibel, 1996), any learned baseform that matched a word pronunciation currently in the dictionary was discarded. However, the homophones are an inherent characteristic of spoken language and discarding them can severely harm the ASR performance. Thus, a more sophisticated way of handling the effects of pronunciations in the system's confusability is needed.

Another effort to constrain the confusability is presented in (Tsai et al., 2007). It is claimed that a pronunciation frequently occurring in many other words introduces extra confusability and hence its importance should be limited. To do so, the inverse word frequency (iwf) is used as a pruning factor to determine the appropriate number of pronunciations to be included in the dictionary. In (Williams and Renals, 1998), an investigation is presented on the use of confidence measures for the evaluation of pronunciation models and on the use of these evaluations in an automatic baseform learning process. The goal when evaluating a pronunciation model for some word is to determine how well the model matches acoustic realizations of that word and an acoustic confidence measure is suitable for such a task, as it can be defined as a function which quantifies how well a

model matches some acoustic data. Another way to reduce the number of pronunciation variants is presented in (Hain, 2005), where the variant with the highest frequency of occurrence in the training data is kept after forced alignment. In (Svendsen et al., 1995), a data-driven optimization of the pronunciation dictionary is proposed, which creates a single baseform per word, subject to a maximum-likelihood criterion.

A method with interesting results in the directions of constraining the confusability of an ASR system is the discriminative training of its different parts. Discriminative training has been successfully applied for acoustic modeling training (Povey, 2003) and for language modeling training (Roark et al., 2004). In the pronunciation modeling training recently some efforts have been made. There are some proposed methods of choosing pronunciations that are directly correlated with the recognition performance. In (Vinyals et al., 2009) and in (Adde et al., 2010) the aim is to find the variant that minimizes the risk of introducing recognition errors. To do so, they adapt the Minimum Classification Error (MCE) framework. Most of such works however are tested on small corpora consisting of short sequences (or even on isolated word recognition tasks) and using small pronunciation dictionaries covering few pronunciation variants. When trying to implement these methods to real data, there are often computational problems and it is not sure these methods are always well generalizable.

2.2.3 Speech-dependent lexicons

There are efforts made in the direction of constructing lexicons that will constrain the confusability caused by the recognition lexicon. One such effort is the construction of speech-dependent lexicon, adapted to the data, ideally with weights suitably trained. To do so, the FST representation of the data is proven to be efficient (for more details on the FSTs see Section 2.3). However, other representations of the phonemic sequences are also possible. In (Chen, 2011), for example, the reference and surface phones are aligned using an HMM representation. Most of these methods make use of a suitable way to generate the uttered phoneme sequence, align it with the reference sequence and find the surface (spoken) pronunciations that correspond to the baseform pronunciations. These methods are of course limited to words present in the training set.

To circumvent this limitation, it is also possible to extract phonological rules once the alignment is done. These rules are not the result of linguistic knowledge as the ones used in knowledge-based approaches (see Section 2.1). It is not even sure they correspond to any linguistic phenomena. They just adapt the baseform pronunciations to a transcription that better matches the spoken utterance. These rules can better represent a particular speaker or even compensate for errors of the ASR system. Some examples of such approaches are given in (Cremelie and Martens, 1999), (Riley et al., 1999), (Yang et al., 2002), (Akita and Kawahara, 2005) and (Van Bael et al., 2007).

Ideally, the observed transcriptions of spoken utterances are obtained manually by listening to these utterances and by writing down the corresponding phonetization, called the auditorily verified (AV) transcription. Unfortunately, collecting AV transcriptions for a large and representative set of training utterances is very costly. This explains why only a few studies (e.g. (Riley et al., 1999)) actually use such transcriptions. A more

practical approach is to make use of the acoustic models to retrieve the most likely pronunciation and to consider this pronunciation as the observed one. In that case one can use a phoneme recognizer as already mentioned (i.e. (Weintraub et al., 1996), (Wolff et al., 2001)). It is desirable that the phoneme recognizer incorporates n-gram phonotactics. Using a phoneme-loop recognizer that imposes no constraints can lead to many errors -especially if the phoneme recognizer is of low quality- that are difficult to recover. Alternatively, an aligner can be used that lines up the utterance with an automaton representing the baseline transcription and some deviations (phoneme deletions, insertions, substitutions) thereof, like in (Cremelie and Martens, 1999), (Riley et al., 1999), (Yang et al., 2002).

Such phoneme alignments that allow the extraction of phonological rules, have been shown to be particularly useful for the recognition of non-native speech. Some examples are given in (Amdal et al., 2000), (Nakamura et al., 2002), (Ward et al., 2002), (Goronzy and Kompe, 2004) and (Huang et al., 2004).

Once these rules are learned, the next step should be to suitably train their weights. The aim of the training is to enforce the discriminative power of the added pronunciation variants keeping the introduced confusability low. In a lot of the works mentioned above, no training of the weights is effected. In (Shu and Lee Hetherington, 2002), a method for training an alignment model represented as an FST with the EM algorithm is introduced. This method is tested on the extraction of phonological rules and it is shown that the Word Error Rate (WER) can be reduced. It is also shown that weighting word-dependent phonemic pronunciations reduce WER more than using weighting phonological rules. However, a trained set of phonological rules has the advantage that is can provide pronunciation weights also for unseen words. Another EM training of the weights of the lexicon based on the utterances of a given word is presented in (Hazen et al., 2005) and (Badr et al., 2010). A discriminative training of the weights might be better generalizable and applicable to phonological rules, as EM training has the drawbacks of finding a local maximum and of often over-fitting to the training data.

2.2.4 Combining g2p conversion and speech-dependent lexicons

An interesting idea of lexicon enhancement was presented in (Beaufays et al., 2003). Their procedure works by initializing a hypothesis with a g2p converter and thereafter refining it with hypotheses from the joint alignment of phone lattices obtained from audio samples and the reference transcriptions. Good results are shown for proper name recognition using this method. Another example of such lexicon enhancement combining g2p conversion and automatic transcriptions is presented in (Choueiter et al., 2007). Thus, transcribed utterances can be used to correct a lexicon generated by a g2p conversion, which is prone to errors especially for low-frequency and irregular words. This idea is developed also in (Bodenstab and Fanty, 2007) using a multi-pass algorithm combining audio information and g2p conversion. During the first pass, audio samples are analyzed and frequent phonetic deviations from the canonical pronunciation (generated previously by a g2p converter) are derived. The second pass then constrains the set of possible pronunciation variations, and forces each audio sample to “choose” which pronunciation best

represent its acoustics.

Apart from improving a g2p-generated lexicon using audio data, another good reason for trying to learn lexicons with such combined methods is the cost issue as mentioned in (Goel et al., 2010). If each time we need to develop a pronunciation lexicon for a language it is necessary to call upon qualified phoneticians, it can be quite expensive and time consuming. In addition, if more than one human is involved there may be differences in opinion. Moreover, human mistakes cannot be avoided. And to reduce in a satisfying way the OOV rate, a large dictionary is needed. What is proposed in (Goel et al., 2010) is to bootstrap learning with a limited lexicon consisting of the 1000 most frequent words in the training data. The next step is to train a g2p system using this lexicon. This g2p system is then used to generate multiple pronunciations for all words used for building the LM which covers the test data. The next step is to use the training data to select the best pronunciation out of these multiple pronunciations via a forced alignment. Then, a new lexicon is created with the selected pronunciations, the acoustic models are retrained and the above process is iterated. The new lexicon is however limited to words seen in the training data, as in all methods where no phonological rules are extracted.

In the former paragraph, it is noted that in (Goel et al., 2010) the acoustic models are retrained in each iteration of their algorithm. The question of how to incorporate pronunciation variability into acoustic modeling warrants a further discussion. Interestingly, it has been found that straightforward approaches to this problem often fail. One possible approach would be to train a pronunciation model, use this pronunciation model to retranscribe the acoustic training set to obtain a surface form transcription and retrain the acoustic models. However, as has been discussed by (Saraclar and Khudanpur, 1999), this can lead to degradation in ASR performance. In (Saraclar and Khudanpur, 1999), they demonstrate that when a base phone is realized as a surface-form, the acoustic model should model it as such, meaning it should retain dependencies on both baseform and surface-form phones. A solution proposed in (Venkataramani and Byrne, 2001) is to align the baseform with the surface transcription, construct a hybrid transcription and use it in supervised MLLR adaptation. In this thesis, the acoustic models will not be retrained each time the pronunciation lexicon changes. The acoustic models that are used are large models with context which cover most cases of phonemic variation. However, as a future work plan, the acoustic models could be retrained with the new pronunciation lexicons in order to have a better consistency in modeling phonemic variation throughout the different parts of the ASR system.

To sum up, in this work we support the idea that it is better to model phonemic variation and confusability at a sub-word level to avoid the problem of word sparseness and allow generalization to unseen data. In particular, we chose phonemes as the modeling unit, in a first place for practical reasons; available dictionaries use this unit and it is a unit more easily manageable than syllables in several languages and more precisely in English, that is the language we mainly focus on in this thesis. Concerning confusability, most works do not take into account how the pronunciation model influences the whole decoding process. In addition, it is difficult to model confusability with a measure that presents a high correlation with the error rate of the system.

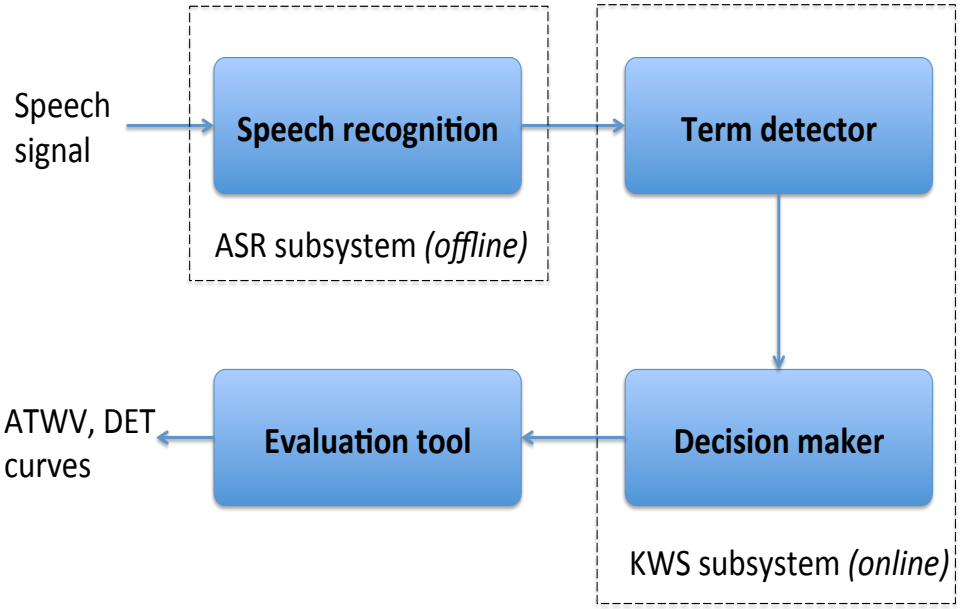

Figure 2.1: *Basic KWS architecture*

2.2.5 Phonemic confusability in the Keyword-Spotting task

As the available data archives (audio, multimedia,...) increase constantly, efficient ways to index and search them become crucial in order to provide access to them. Keyword spotting (KWS) aims at detecting search terms in spoken utterances. Given a user-defined search term, the keyword spotter predicts the best time span for this word in the spoken utterance along with a confidence score. If the prediction confidence is above a certain level, the keyword is declared to be spoken in the utterance within the predicted time span, otherwise the keyword is declared as not occurring in the speech segment. The problem in keyword spotting training is directly related to the confusability problem, as its desired performance is a highly discriminative one. The model parameters are indeed chosen so that the utterance in which the keyword is spoken would have a higher confidence than any other spoken utterance in which the keyword is not spoken.

The standard KWS architecture consists of an ASR subsystem to produce the word/subword lattices and a KWS subsystem for term detection. The ASR system could also generate 1-best, n-best or confusion networks of words/subwords. In this thesis, we will work with lattices. The speech recognition typically generates the lattices from the speech signal; a term detector searches these lattices for putative occurrences of the search terms; a decision maker finally decides whether each putative detection is reliable. This basic architecture can be seen in Figure 2.1.

Since this is a detection task, performance can be characterized by Receiver Operating Characteristic (ROC) curves of miss (Pmiss) versus false alarm (Pfa) probabilities, or

by a weighted function of the two probabilities. For the NIST STD06 evaluation the primary evaluation metric was the actual term-weighted value (ATWV), which is defined as follows (NIST, 2006).

$$ATWV = 1 - \frac{1}{T}\sum_{t=1}^{T}\left(P_{miss}(t) + \beta P_{fa}(t)\right) \tag{2.1}$$

$$P_{miss}(t) = 1 - \frac{N_{corr}(t)}{N_{true}(t)}, P_{fa}(t) = \frac{N_{spurious}(t)}{Total - N_{true}(t)} \tag{2.2}$$

where T is the total number of terms, β is set to approximately 1000, N_{corr} and $N_{spurious}$ are the total number of correct and spurious (incorrect) term detections, N_{true} is the total number of true term occurrences in the corpus, and $Total$ is the duration (in seconds) of the indexed audio corpus. For a detailed definition of the ROC curves, see Section 6.3.1.

To be more accurate, the term detection component of this architecture is developed in two phases, the index and the search phase. The index creation is done offline and is independent of the search terms. Its purpose is to create a suitable database structure that will allow the fast and robust detection of spoken utterances of a particular term during the search phase. The search of a particular term is an online procedure.

Creating the index from the 1-best word recognition result is the simplest and fastest way to do it. However, in this way a recognition mistake concerning a search-term word leads directly to a detection error, either a miss or a false alarm (FA). The problem becomes of course more acute if the used ASR system is weak, which can often be the case for low-resourced languages, for noisy data, etc. Thus, to avoid misses, a solution is to use the whole hypothesized word lattice.

There are different ways of creating the index. In (Vergyri et al., 2007), the authors propose the use of an n-gram index with scores and time information. The score of each n-gram is the posterior probability, computed as the forward-backward combined score (acoustic, language and prosodic scores were used) through all the lattice paths that share the n-gram nodes. In (Allauzen et al., 2004) a deterministic weighted finite-state transducer is constructed as index, storing soft-hits in the form of (utterance ID, start time, end time, posterior score) quadruplets. In (Wallace et al., 2011) the index takes the form of a phonetic posterior-feature matrix.

An important problem in KWS is the problem of OOVs, and not only because an OOV is directly an error in the KWS system. Their importance depends on the evaluation measure used. The ATWV measure has two characteristics: (1) missing a term is penalized more heavily than having a false alarm for that term, (2) detection results are averaged over all query terms rather than over their occurrences, i.e. the performance metric considers the contribution of each term equally. Thus OOVs, though infrequent, have the same impact on the metric as in-vocabulary (IV) terms (Akbacak et al., 2008). Moreover, the impact of OOVs in real life can be important as they are often related to names and to important/new events. The problem is serious: for instance, in the study of (Logan et al., 2000), it is found that about 12% of the users queries contain OOV words.

A possible solution for handling the problem of OOVs and recognition errors is the use of hybrid systems. In the hybrid systems, rather than using the straightforward solution

of a word index, different sub-word units are employed to perform the term detection and confidence estimation. Typically, a phoneme-based system is used to handle OOV terms. In this approach, search terms are converted to pronunciations by g2p models, and the pronunciations are searched for in a phoneme lattice generated by a speech recognizer (see (Saraclar and Sproat, 2004) and (Can et al., 2009) among others).

Sometimes, some sub-word units are used for detection and others for estimating the confidence. For example, in (Tejedor et al., 2009) the best system is found when phonemes are used as detection units, while graphemes are used for confidence estimation. Another interesting idea in this paper is that not only system hybridization is used, but also system combination. For instance, it is possible to combine a phoneme-based and a grapheme-based system using the same confidence estimation. This kind of system combination is shown to improve the system performance for English and Spanish. Especially for English, which is the language we experiment on in this thesis, more significant improvement is observed. In English, it is known that the pronunciation rules are rather complex, leading to an irregular relationship between graphemes and phonemes, which makes the grapheme- and phoneme-based systems capture different information of the language and exhibit complementary performance on KWS. That is why in the work presented in this thesis (see Chapter 6) we decided to build an English phoneme-based KWS system and try to improve it. A future aim is indeed to integrate it in a hybrid system. The current state-of-the-art KWS system are mainly hybrid systems as they perform better even for IV terms.

Another approach to compensate for OOVs in a hybrid system is the one presented in (Bisani and Ney, 2005), where subword units called graphones are used. Each graphone is a pair of a letter sequence and a phoneme sequence of possibly different lengths. Graphones are trained using a pronunciation dictionary and are used to replace the OOVs.

There are many challenges in finding a good operating point for a KWS system that balances false alarms and true hits, particularly when the queries are OOV terms. OOV terms have more unpredictable pronunciations, caused by ASR errors, pronunciation variation, and acoustic variation. A solution is to use n-best predictions in pronunciations (Wang et al., 2009). However, this increases the FA rate, which is why a confidence pruning threshold is set on the predictions. There are different ways of setting this confidence threshold. In (Wang et al., 2009), it is proposed to integrate the confidence of the predicted pronunciation with the confidence of a term detection so that a jointly-optimal postponed decision rule can be formulated based on a compound confidence. In (Parada et al., 2010), it is shown that automatically tagging OOV regions helps to reduce false alarms. To detect OOVs regions, the authors use a system that combines the posterior probability of subword units and the entropy of the subword unit in the region of interest. An interesting result of this work is that simply increased n-best representation of queries did not translate to significant improvements in KWS performance. Instead, incorporating phonetic confusability increased the hits by helping to compensate for potential differences in deriving index and query representations.

A confusion matrix can directly compensate for differences between the index and the query representations. This idea was first introduced in spoken document retrieval in (Moreau et al., 2004). In (Zhang et al., 2006) the keywords are expanded using a phoneme

confusion matrix. However, only substitutions are taken into account, and no training of the matrix weights is effected. An improved version of this method is presented in (Wang and Zhang, 2012), where the confusion matrix is combined with a word-level minimum classification error training method.

Another possible solution is to incorporate phonemic confusions directly in the acoustic model. In (Pinto et al., 2007) an hybrid Markov model-artificial neural network (HMM-ANN) keyword system is used and an acoustic confusion model is built such as to take into account the systematic errors made by the neural network. In (Wallace et al., 2011) the phoneme confusions are directly incorporated in the index, as an index is created not of discrete phone instances but rather of probabilistic acoustic scores of these phones at each time instant. Thus, the indexing phase produces a posterior-feature matrix. The search phase is performed on this matrix by calculating the likely locations of term occurrences through estimation of their likelihood from the probabilistic scores.

2.3 FST background

2.3.1 Generalities

In the last two decades, FSTs have been shown to be useful for a number of applications in speech and language processing (Mohri, 1997). FST operations such as composition, determinization, and minimization make manipulating FSTs both effective and efficient. In this thesis, all the FST manipulations are realized using the OpenFst library (Allauzen et al., 2007).

Weighted transducers (resp. automata) are finite-state transducers (resp. automata) in which each transition carries some weight in addition to the input and output (resp. input) labels. The interpretation of the weights depends on the algebraic structure of the semiring in which they are defined.

2.3.2 Semiring

A *semiring* is a system $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ containing a set of weights $\mathbb{K}$ and the operators $\oplus$ and $\otimes$, such that: $(\mathbb{K}, \oplus, \bar{0})$ is a commutative monoid with $\bar{0}$ as the identity element for $\oplus$; $(\mathbb{K}, \otimes, \bar{1})$ is a monoid with $\bar{1}$ as the identity element for $\otimes$; $\otimes$ distributes over $\oplus$: for all a, b, c in $\mathbb{K}$: $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$ and $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$, and $\bar{0}$ is an annihilator for $\otimes : \forall a \in \mathbb{K}, a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$. When manipulating weighted transducers, the $\otimes$ and $\oplus$ operators are used to combine weights in a serial and parallel fashion, respectively. A semiring is commutative when $\otimes$ is commutative.

2.3.3 Weighted Finite-State Transducers

A *weighted finite-state transducer* T over a semiring $\mathbb{K}$ is an 8-tuple $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ where: Σ is the finite input alphabet of the transducer; Δ is the finite output alphabet; Q is a finite set of states; $I \subseteq Q$ the set of initial states; $F \subseteq Q$ the set of final

states; $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times Q$ a finite set of transitions; $\lambda : I \to \mathbb{K}$ the initial weight function; and $\rho : F \to \mathbb{K}$ the final weight function mapping F to $\mathbb{K}$.

A *weighted automaton* $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ is defined in a similar way simply by omitting the output labels. The weighted transducers and automata considered in this work are assumed to be trimmed, i.e. all their states are both accessible and co-accessible. Omitting the input labels of a weighted transducer T results in a weighted automaton which is said to be the output projection of T.

Using the notation of (Cortes et al., 2006), if $e = (q, a, b, q', w)$ is a transition in E, $p(e) = q$ (resp. $n(e) = q'$) denotes its origin (resp. destination) state, $i(e) = a$ its input label, $o[e] = b$ its output label and $w(e) = E(e)$ its weight. These notations extend to paths: if π is a path in T, $p(\pi)$ (resp. $n(\pi)$) is its initial (resp. ending) state and $i(\pi)$ is the label along the path. We denote by $P(q, q')$ the set of paths from q to q' and by $P(q, x, y, q')$ the set of paths for q to q' with input label $x \in \Sigma^*$ and output label $y \in \Delta^*$. The path from an initial to a final state is a successful path. The weight associated by a weighted transducer T to a pair of strings $(x, y) \in \Sigma^* \times \Delta^*$ is denoted by $T(x, y)$ and is obtained by $\otimes$-summing the weights of all successful paths with input label x and output label y:

$$T(x, y) = \bigoplus_{\pi \in P(I,x,y,F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi]) \tag{2.3}$$

$T(x, y) = \bar{0}$ when $P(I, x, y, F) = \emptyset$.

Another notion that will be used in the KWS chapter (Chapter 6) is the notion of *factor automaton*. A factor automaton is a finite automaton which accepts the set of all sub-strings (factors) of the string. It is an efficient way to represent the set of all sub-strings where each sub-string corresponds to exactly one path. Thus, any matching string can be found in time equal to the length of the pattern looked for.

2.3.4 Some useful semirings

The *real semiring* $(\mathbb{R}, +, \times, 0, 1)$ is a suitable semiring to use when the weights represent probabilities. The *log semiring* is defined as $(\mathbb{R} \cup [-\infty, \infty], -\log(\exp(-x) + exp(-y)), +, \infty, 0)$. It is isomorphic to the real semiring via the negative-log mapping and is used in practice for numerical stability. Another commonly used semiring is the *tropical semiring*, which is the standard semiring in OpenFst. It is defined as $(\mathbb{R} \cup [-\infty, \infty], min, +, \infty, 0)$, i.e., min is the $\oplus$ of the semiring (with identity ∞) and $+$ is the $\otimes$ of the semiring (with identity 0). The log and the tropical semirings are of particular importance for speech processing.

In Chapter 6, reference is frequently made to special semiring structures defined on the Cartesian product of ordered sets. More precisely, the index structure of the presented KWS system is a weighted transducer over the lexicographic semiring. We will give here the definitions of the product semiring and the lexicographic semiring, as the latter is based on the definition of the former. The *product semiring* of two partially-ordered semirings $\mathcal{A} = (\mathbb{A}, \oplus_{\mathbb{A}}, \otimes_{\mathbb{A}}, \bar{0}_{\mathbb{A}}, \bar{1}_{\mathbb{A}})$ and $\mathcal{B} = (\mathbb{B}, \oplus_{\mathbb{B}}, \otimes_{\mathbb{B}}, \bar{0}_{\mathbb{B}}, \bar{1}_{\mathbb{B}})$ is defined as

$$\mathcal{A} \times \mathcal{B} = (\mathbb{A} \times \mathbb{B}, \oplus_{\times}, \otimes_{\times}, \bar{0}_{\mathbb{A}} \times \bar{0}_{\mathbb{B}}, \bar{1}_{\mathbb{A}} \times \bar{1}_{\mathbb{B}}) \tag{2.4}$$

where $\oplus_\times$ and $\otimes_\times$ are component-wise operators, e.g. $\forall a_1, a_2 \in \mathbb{A}, b_1, b_2 \in \mathbb{B} : (a_1, b_1) \oplus_\times (a_2, b_2) = (a_1 \oplus_\mathbb{A} a_2, b_1 \oplus_\mathbb{B} b_2)$. A *partial order* on a semiring $\mathbb{K}$ is a relation $<_\mathbb{K}$ defined as: $\forall x, y \in \mathbb{K}, x <_\mathbb{K} y \Longleftrightarrow x = x \oplus y$.

The *lexicographic semiring* of two partially-ordered semirings $\mathcal{A} = (\mathbb{A}, \oplus_\mathbb{A}, \otimes_\mathbb{A}, \bar{0}_\mathbb{A}, \bar{1}_\mathbb{A})$ and $\mathcal{B} = (\mathbb{B}, \oplus_\mathbb{B}, \otimes_\mathbb{B}, \bar{0}_\mathbb{B}, \bar{1}_\mathbb{B})$ is defined as

$$\mathcal{A} * \mathcal{B} = (\mathbb{A} \times \mathbb{B}, \oplus_*, \otimes_*, \bar{0}_\mathbb{A} \times \bar{0}_\mathbb{B}, \bar{1}_\mathbb{A} \times \bar{1}_\mathbb{B}) \tag{2.5}$$

where $\otimes_*$ is component-wise multiplication operator and $\oplus_*$ is a lexicographic priority operator, $\forall a_1, a_2 \in \mathbb{A}, b_1, b_2 \in \mathbb{B}$:

$$(a_1, b_1) \oplus_* (a_2, b_2) = \left\{ \begin{matrix} (a_1, b_1) & if \quad a_1 = a_1 \oplus_\mathbb{A} a_2 \quad or \quad (a_1 = a_2 \quad and \quad b_1 = b_1 \oplus_\mathbb{B} b_2) \\ (a_2, b_2) & otherwise \end{matrix} \right\} \tag{2.6}$$

Using the definition of the partial order given above, the $\oplus_*$ operator can be defined as:

$$(a_1, b_1) \oplus_* (a_2, b_2) = \left\{ \begin{matrix} (a_1, b_1) & if \quad a_1 <_\mathbb{A} a_2 \quad or \quad (a_1 = a_2 \quad and \quad b_1 <_\mathbb{B} b_2) \\ (a_2, b_2) & otherwise \end{matrix} \right\} \tag{2.7}$$

More generally, one can define the product and lexicographic semirings on the Cartesian product of n ordered sets $\{\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_n\}$. In addition, the product and lexicographic semirings on $\{\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_n\}$ can be recursively defined using the associativity of $\times$ and $*$ operators. For example for the product semiring:

$$\mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n = ((\cdots (\mathcal{A}_1 \times \mathcal{A}_2) \times \cdots) \times \mathcal{A}_n). \tag{2.8}$$

2.3.5 Algorithms

This section presents several fundamental algorithms for weighted transducers and automata. For more details, the reader is referred to (Mohri, 2009).

The *Composition* operation allows different levels of information to be combined. The composition $T = T_1 \circ T_2$ of two transducers T_1 and T_2 has exactly one path mapping sequence x to sequence y for each pair of paths, the first in T_1 mapping x to some sequence z and the second in T_2 mapping z to y. The weight of a path in T is the $\otimes$-product of the weights of the corresponding paths in T_1 and T_2 (Salomaa and Soittola, 1978). Its formal definition can be given as:

$$T(x, y) = T_1 \circ T_2 = \bigoplus_z T_1(x, z) \otimes T_2(z, y), \tag{2.9}$$

when the semiring is commutative and the sum $\bigoplus$ is well-defined and in $\mathbb{K}$ for all x, y.

Another useful operation is the *Determinization*. An automaton is deterministic if it has a single initial state, there are no two arcs leaving a state with the same input label and there are no input epsilon labels. Determinization takes an FST and attempts to construct a deterministic equivalent. Weighted determinization requires some technical conditions on the semiring or the weighted automaton. The algorithm works with any weakly divisible

semiring. A semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ is said to be weakly divisible if for any x and y in $\mathbb{K}$ such that $x \oplus y \neq \bar{0}$, there exists at least one z such that $x = (x \oplus y) \oplus z$. Additionally, it is assumed that for any string $x \in \Sigma^*$, the sum of the weights of the paths labeled with x and starting at an initial state is non-$\bar{0}$: $w[P(I, x, Q)] \neq \bar{0}$. This condition is always satisfied with trim weighted automata over the tropical semiring or any zero-sum-free semiring. Determinization does not terminate for all weighted automata. There exists, however, a general twins property for weighted automata that provides a characterization of determinizable weighted automata under some general conditions (see (Mohri, 2009) for more details).

Minimization takes a deterministic WFSA and returns an equivalent WFSA with the minimal amount of states. It first pushes the weights in the WFSA and then encodes it as a costless acceptor. In speech recognition, it can make a size reduction in the used transducers.

Concerning the *weight pushing*, which is done implicitly in minimization as noted earlier, weights can be pushed towards the initial or the final state. Pushing is a special type of reweighting which globally preserves the path weights. Weight pushing is used when the *normalization* of the weights of a WFST is desired. This algorithm can thus be used in order to make an FST stochastic, meaning that the weights of all arcs leaving a state sum to one. Normalization is particularly important because it enables the use of log-linear parameterizations. It enables to transform the weights to joint or conditional probabilities and we will use it later in this thesis in the context of discriminative training.

It should be noted that the normalization realized by weight pushing is a global, and not a per-state one. As explained by (Eisner, 2002), the per-state normalization is a more constrained one. In fact, the result may differ for equivalent FSTs that express the same weighted relation. Undesirable consequences of this fact have been termed "label bias" (Lafferty et al., 2001). Also, in the conditional case such per-state normalization is only correct if all states accept all input suffixes (since "dead ends" leak probability mass). In the joint case, this problem can be cirmumvented by pruning all non-coaccessible states before normalization.

Another category of algorithms that interests us are the ones related to *Shortest distance* and *Shortest path* problems. A general definition of shortest distance from a state q to a state p is the sum of the weights of all paths from q to p using $\oplus$ as already mentioned earlier. In this work, the shortest distance is particularly useful for the calculation of the forward and backward probabilities. Let T be a weighted transducer over a semiring $\mathbb{K}$. For any state $q \in Q$, we denote by $d[q]$ the shortest distance from I to q (forward probability on the real or the log semiring) and by $f[q]$ the shortest distance from q to F (backward probability in the real or the log semiring):

$$d[q] = \bigoplus_{\pi \in P(I,q)} (\lambda(p[\pi]) \otimes w[\pi]) \tag{2.10}$$

$$f[q] = \bigoplus_{\pi \in P(q,F)} (w[\pi] \otimes \rho(n[\pi])) \tag{2.11}$$

The shortest path algorithm (resp. n-shortest path) computes the 1-best (resp. n-best) paths in the transducer T. The shortest distance is defined for any semiring k-closed to T.

Let $k >= 0$ be an integer. A semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ is said to be k-closed if

$$\forall x \in \mathbb{K}, \bigoplus_{n=0}^{k+1} x^n = \bigoplus_{n=0}^{k} x^n. \tag{2.12}$$

For instance, the tropical semiring is a k-closed semiring with $k = 0$.

To calculate the shortest path, the semiring must have the path property $a \oplus b \in a, b$. For instance, the tropical semiring, where the $\oplus$ is the minimum operator, which actually computes a shortest path.

2.3.6 Entropy semiring

Next, a detail description of the expectation semiring is given and its use to calculate entropy is explained. It will be a necessary concept for our work presented in Chapter 4.

The entropy $H(p)$ of a probability mass function p defined over a discrete set X is defined as (Cover and Thomas, 1991):

$$H(p) = -\sum_{x \in X} p(x) log p(x), \tag{2.13}$$

where by convention $0log0 = 0$. This definition can be extended to probabilistic automata which define distributions over sets of strings. We call an automaton probabilistic if for any state $q \in Q$, the sum of the weights of all cycles at q is well-defined and in $\mathbb{K}$ and $\sum_{x \in \Sigma^*} A(x) = 1$. In a probabilistic automaton, the target of a transition is no longer a single state, but is a probabilistic choice over several next states (Paz, 1971). The entropy of A can be written as:

$$H(A) = -\sum_{x} A(x) log A(x), \tag{2.14}$$

where $A(x)$ is the output weight associated by an automaton A to an input string $x \in \Sigma^*$.

The expectation (or entropy) semiring is defined in (Eisner, 2001) as $(\mathbb{K}, \oplus, \otimes, (0, 0), (1, 0))$, where $\mathbb{K}$ denotes $(\mathbb{R} \cup [-\infty, \infty]) \times (\mathbb{R} \cup [-\infty, \infty])$. For weight pairs (a_1, b_1) and (a_2, b_2) in the real semiring, the $\oplus$ and $\otimes$ operations are defined as follows [1]:

$$(a_1, b_1) \oplus (a_2, b_2) = (a_1 + a_2, b_1 + b_2) \tag{2.15}$$

$$(a_1, b_1) \otimes (a_2, b_2) = (a_1 a_2, a_1 b_2 + a_2 b_1) \tag{2.16}$$

The entropy of A defined in equation (2.14) can be computed as a single-source shortest distance for an automaton defined over the entropy semiring (Cortes et al., 2006) with weights $(w, -wlogw)$ where w is in $\mathbb{R}$. If the sum of the weights of all paths from any state $p \in Q$ to any state $q \in Q$ is well-defined, the shortest distance from p to q is:

$$d[p, q] = \bigoplus_{\pi \in P(p,q)} w[\pi]. \tag{2.17}$$

1. Usual conventions apply: $a \oplus \infty = a$, $a \otimes \infty = \infty$.

Thus, the shortest distance from the initial states to the final states for the probabilistic automaton A with weights $(w, -wlogw)$ in $\mathbb{K}$ will be:

$$d[I, F] = (\sum_x A(x), -\sum_x A(x) log A(x)) = (1, H(A)). \tag{2.18}$$

2.3.7 Matchers

We will now make a special reference to matchers provided by OpenFst because, as we will see later, they will be necessary in this work. Matchers can find and iterate through requested labels at FST states; their main use is in composition matching. In the simplest form, these are just a search or hash keyed on labels. More generally, they may implement matching special symbols that represent sets of labels such as ρ (rest), σ (all) or ϕ (fail), which can be used for more compact automata representations and faster matching. In Table 2.1 a summary is given of the basic matchers and their effect.

Matcher	Matches	Consumes
ϕ	Rest	None
ρ	Rest	All
σ	All	All
ϵ	All	None

Table 2.1: *Basic matchers*

2.3.8 FST-based speech recognition

In a speech recognition system, each knowledge source can be represented as an FST (Mohri et al., 2002). Typically, this concerns the language model, the pronunciation lexicon, the context dependency and the acoustic model. These parts are combined using the composition operation which allows the combination of different levels of representation. Then some optimizations are performed (remove epsilons, determinization and minimization) and the final step is the decoding. The decoding can be implemented by the shortest path algorithm on the tropical semiring. In fact, the tropical semiring is appropriate for performing Viterbi search using negative log probabilities: we add negative logs along a path and take the min between paths. An abstract presentation of the decoding process can be given as:

$$\hat{W} = bestpath(H \circ C \circ L \circ G), \tag{2.19}$$

where $\hat{W}$ is the sequence of words corresponding to the best recognition hypothesis. H represents the HMM set and C is an FST that maps the context-dependent to context-independent phonemes. L is the pronunciation model FST, containing a mapping of sequences of phonemes to words, G is the language model finite state automaton (FSA), which contains n-gram statistics, and $\circ$ is the composition operator.

The FSTs provide a common and natural representation of these models and general transducer operations combine them flexibly and efficiently. Weighted determinization

and minimization algorithms optimize their time and space requirements, and a weight pushing algorithm distributes the weights along the paths of a weighted transducer optimally for speech recognition. Thus, the FST approach has recently become a state-of-the-art technology in speech recognition allowing the implementation of large models reducing decoding time.

Chapter 3

Pronunciation and pronunciation variants generation using SMT-inspired approaches

In this chapter two alternative methods are proposed, inspired by machine translation, to derive pronunciations and pronunciation variants from an initial lexicon. First, a machine translation tool is used as a g2p converter to extract an n-best pronunciation list. The same toolkit is used to perform p2p conversion and derive variants from a given canonical pronunciation. The second approach is a novel method based on a pivot approach, previously used for the paraphrase extraction task, and here applied as a post-processing step to the g2p converter or directly to the canonical pronunciation as a p2p converter. Our g2p converter allows to generate pronunciations for OOVs, but also enriches our original dictionary with pronunciation variants, while the p2p converters focuses on adding variation to the dictionary. The performance of these two methods is tested under different training conditions (using for the training a lexicon with or without alternative pronunciations). The recall and precision results on the g2p and p2p conversion tasks prove the generation of good quality pronunciations. Finally some speech recognition experiments are presented using the new, enriched lexicons.

3.1 Introduction

This work aims to generate pronunciation variants in an automatic and language independent way, even when no variants are included in the lexical resources available for training. Most available dictionaries contain no or few variants, or their variants are not consistent or suitable for training. The proposed methods are tested for English, a language known to be difficult for pronunciation generation, since there is a loose relationship between letters and sounds. In addition, there are a lot of words of foreign origin that keep the pronunciation of their initial language, and others that have a variety of pronunciations depending upon the speakers.

Moses (Koehn et al., 2007), a publicly available phrase-based statistical machine translation method is first used as a g2p converter. Then, two options are explored to

generate pronunciation variants. In the first one, variants are derived from the generated n-best list. Moses is also used as a p2p converter to generate variants from baseform pronunciations when variants are included in the training set. However, this method fails when a single-pronunciation dictionary is used for training. In this latter case a novel approach, originally used for the paraphrase extraction task, was proposed to use also graphemic information to generate variants. This novel approach is based on the principle that sequences of modified phonemes can be identified using a graphemic sequence as a pivot. This means that sequences of modified phonemes that correspond to the same graphemic sequence are identified as a plausible variation pattern. This pivot-based method is used to generate variants given a canonical pronunciation of a word. Then, it is used as a post-processing step to the Moses g2p converter, enabling the generation of pronunciations with variants for out-of-vocabulary words (OOVs). It is independent of the origin of the input pronunciations, focusing on local variations, which are the most common pronunciation variants. To the best of our knowledge, this is the first application of such a pivot approach to the generation of pronunciation variants.

The motivation behind using Moses and the pivot method for g2p conversion is the similarity that can be observed between the phrase-based approach in translation and the g2p conversion. This approach allows us to align sub-strings of graphemes to sub-strings of phonemes. Context in both the graphemic and the phonemic transcriptions is taken into account for the pronunciation prediction, which is a desired characteristic for the g2p task. This similarity will further emerge from the description of our system as a machine translation task in the following section.

3.2 Methodology

3.2.1 Moses as g2p and p2p converter

Moses uses a phrase translation model based on the noisy channel model. Following, (Koehn et al., 2003), Bayes' rule is used to reformulate the translation probability for translating a foreign sentence f into (traditionally) English e:

$$\arg\max_e p(e \mid f) = \arg\max_e p(f \mid e)p(e). \tag{3.1}$$

In the g2p conversion task, we replace f with the graphemic representation of a word and e with its phonemic representation. The former equation allows the use of a language model (LM) $p(e)$, which will be a phoneme-based LM in our case, and of a separate "translation model" $p(f|e)$. It expresses the best predicted pronunciation given a word.

During decoder, the input word f is segmented into a set of I grapheme (i.e., letter) sequences ("phrases") $\overline{f}_1^I$. Each letter sequence $\overline{f}_i$ in $\overline{f}_1^I$ is transcribed ("translated") to a phoneme sequence $\overline{e}_i$. A probability distribution $\phi(\overline{f}_i \mid \overline{e}_i)$ models the transcription of these segments. The corresponding grapheme and phoneme sequences with their probabilities are presented in a so called "phrase table". Recall that due to Bayes' rule, the transcription direction is inverted from a modeling standpoint. Thus, $p(f \mid e)$ is decom-

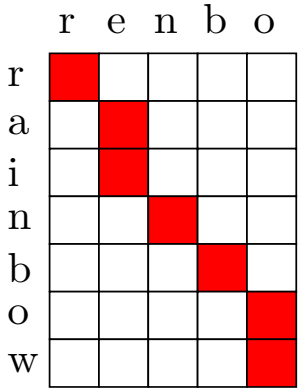

Figure 3.1: *Alignment of the word "rainbow" with its pronunciation /renbo/*

posed into:

$$p(\overline{f}_1^I \mid \overline{e}_1^I) = \prod_{i=1}^{I} \phi(\overline{f}_i \mid \overline{e}_i). \tag{3.2}$$

An alignment model is needed to extract the aligned graphemic and phonemic sequences and calculate their probabilities $\phi(\overline{f}_i \mid \overline{e}_i)$. We learn sequence alignment using GIZA++ (Och and Ney, 2000), an open-source toolkit proposed for the IBM models (Brown et al., 1993). The IBM models that this toolkit implements allow one-to-one alignments. This problem is remedied with the use of some heuristics. First, the parallel corpus is aligned bidirectionally. This gives two alignments, and an expansion heuristic between the intersection and the union is used to reconcile them(Koehn et al., 2003). In this work, a pronunciation dictionary is used in the place of an aligned bilingual text corpora. The orthographic transcription is considered as the source language and the pronunciation as the target language. The graphemes and phonemes are the units of alignment. In Figure 3.1 an example of an alignment of a word of the training set with its pronunciation can be seen. The reader is referred to Appendix A for a list of the used phoneme set.

A 5-gram phoneme language model (LM), estimated on the pronunciations in the training set using the SRI toolkit (Stolcke, 2002), is used to provide additional phonemic information and corresponds to the target LM in SMT. Finally, the combination of all components is fully optimized with a minimum error training step (tuning) on a development set. The tuning strategy used was the standard Moses training framework based on the maximization of the BLEU score (Papineni et al., 2002).

Moses can output an n-best translation list, that is an ordered list of translations of a source string. The 1-, 2-, 5- or 10-best translations (i.e. pronunciation variants) per word are kept.

The above described models can also be used for a p2p converter, which has, more-

over, higher potential for capturing pronunciation variation phenomena in languages like English, where orthography and pronunciation generally have a looser relationship than in other languages. In this case, the source language and the target language are aligned phonemic transcriptions. For the source side sequences we use the canonical pronunciation (the longest one [1]), for the target side sequences itself and/or its variants depending on their existence or not in the different versions of the training set as presented in the Section 3.3.

3.2.2 Pivot paraphrasing approach

The pivot method applies the work of (Bannard and Callison-Burch, 2005) to the generation of pronunciation variants. Paraphrases are alternative ways of conveying the same information. The analogy with pronunciation variants of a word is easily seen: the different pronunciations being alternate phonemic expressions of the same orthographic information. In our case, the "paraphrases" are phonemic sequences of the phrase table generated by Moses from the word-pronunciation training pairs. For each phonemic sequence in the phrase table, we find all corresponding graphemic sequences and then look back to find what other phonemic sequences are associated with the set of graphemic ones. These phonemic sequences are plausible paraphrases.

The paraphrase probability $p(\overline{e}_2 \mid \overline{e}_1)$ of two phonemic sequences $\overline{e}_1$ and $\overline{e}_2$ is assigned in terms of the phrase table probabilities $\phi(\overline{f} \mid \overline{e}_1)$ and $\phi(\overline{e}_2 \mid \overline{f})$ estimated based on the counts of the aligned graphemic-phonemic phrases. Since $\overline{e}_1$ can be translated as multiple graphemic phrases, we sum over $\overline{f}$ for all the corresponding graphemic entries of the phrase table:

$$\hat{e}_2 = \arg\max_{\overline{e}_2 \neq \overline{e}_1} p(\overline{e}_2 \mid \overline{e}_1) \tag{3.3}$$

$$= \arg\max_{\overline{e}_2 \neq \overline{e}_1} \sum_{\overline{f}} \phi(\overline{f} \mid \overline{e}_1)\phi(\overline{e}_2 \mid \overline{f}) \tag{3.4}$$

An example of a paraphrase pattern in the dictionary is:

discou**nt**ed	diskW**nt**xd	dIskW**n**xd
discou**nt**enance	dIskW**nt**Nxns	dIskW**n**Nxns

The alternative pronunciations differ only in the part that can be realized as either **nt** or **n**, while the rest remains the same. The **nt** and **n** form a paraphrased pair, a common pattern found in many words in the pronunciation lexicon. The pivot method focuses on local modifications observed between variants of a word and is a lot faster than the n-best list generation by Moses-g2p. All occurrences of these paraphrased patterns are substituted in the input pronunciations, which can be either the baseform pronunciations or the 1-best pronunciations of Moses-g2p. The second case allows us to add variants also to OOVs.

At this point, different types of pruning are applied on the generated variants. First, the candidate variants are reranked based on additional phonemic contextual information

1. Most of the variants reflect reduced pronunciations found in casual speech.

expressed by the 5-gram phoneme LM already used by Moses for the g2p conversion. The SRI toolkit served for the reranking. Then, pruning is done based on the length of the paraphrases substituted in the pronunciations. It was experimentally found that the quality of the generated variants improves when only 3- and 4-grams paraphrases are substituted because more context is taken into account throughout the procedure and some confusions are avoided.

The Levenshtein Distance between each pronunciation and its generated variants was then calculated. This measure should not exceed a threshold since the different pronunciations of a word are usually phonemically very close. Pruning with thresholds of 3 (LD3) and 2 (LD2), meaning that all the variants with edit distances greater than 3 and 2 respectively are pruned, were tried. Finally, the 1-, 4- and 9-best pronunciation variants per input pronunciation were kept and merged with the input pronunciations in order to have 2-, 5- and 10-best pronunciations generated and so as to be able to compare these with the n-best lists from Moses g2p.

3.3 Experimental setup

The LIMSI American English pronunciation dictionary, created with extensive manual supervision, serves as basis of this work. The pronunciations are represented using a set of 45 phonemes (Lamel and Adda, 1996) (for a detailed presentation of the phoneme set, see Appendix A). 18% of the words are associated with multiple pronunciations. These mainly correspond to well-known phonemic alternatives (for example the pronunciation of the ending “ization”), and to different parts of speech (noun or verb). The dictionary contains a mix of common words, acronyms and proper names, the last two categories being difficult cases for g2p converters and particular effort has been made to pronounce proper names in text-to-speech synthesis technology (Spiegel, 1993). Case distinction is eliminated since in general it does not influence the word’s pronunciation, the main exceptions being the few acronyms which have a spoken and spelled form. Some symbols in the graphemic form are not pronounced, such as the hyphen in compound words.

The dictionary has 187,975 word entries (excluding words starting with numbers) with on average 1.2 pronunciations per word. Each dictionary entry contains the orthographic form of a word and its pronunciations (one or more). The majority of words have only one pronunciation, leaving it to the acoustic model to represent the observed variants in the training set that are due to allophonic differences. Moreover, since the dictionary is mostly manually constructed, it is certainly incomplete with respect to coverage of pronunciation variants particularly for uncommon words. The pronunciations of words of foreign origin (mostly proper names) may also be incomplete since their pronunciation depends highly on the speaker’s knowledge of the language of origin. This means that some of the automatically generated variants are likely to be correct (or plausible) even if they are not in the current version of the Master dictionary. It was however decided to use this dictionary as it is reputed to be a high quality dictionary for speech recognition,

which will be the domain of application of the proposed methods in Section 3.5 [2].

The dictionary was randomly split based on the graphemic form of the word into a training, a development (dev) and a test set. The dev set is necessary for the tuning of the Moses model. In order to have a format that resembles the aligned parallel texts used for training machine translation models, the dictionary is expanded so that each entry corresponds to a word-one pronunciation pair. The resulting dev and test sets have 11k and 19k distinct entries.

The g2p converter is trained for two conditions, on the entire training subset using all pronunciations of words (tr_set), or on the same word list but using only one (canonical) pronunciation per word (tr_set_l). Since canonical pronunciations are not explicitly indicated in the lexicon, the longest one is taken as the canonical form. In the first training condition, there are 200k entries (distinct word-pronunciation pairs) in the training set with on average 1.2 pronunciations/word. In the second training condition, the training set has 160k entries with a single pronunciation per word.

Concerning the p2p systems, three different training conditions are compared. The first two are the ones already presented for the g2p training in the previous paragraph. The third training condition concerns training only on words with two or more pronunciation variants. Thus, all words in this training set have multiple pronunciations (tr_set_m).

At this point, a further preparation of the training set for each method is required. For the method where Moses is used as a p2p converter, a "monolingual" parallel corpus is needed, meaning that both the source language and the target language will have phonemes as elements. The source language is always formed by the canonical pronunciation segmented into phonemes. The target language is formed by the corresponding pronunciations depending on the training condition. When Moses is used as a g2p converter as well as for the pivot method, the training set is used as a parallel corpus with one graphemic transcription - pronunciation pair per line with spaces separating characters, in order to generate n-best lists (Moses-g2p converter) or to use the generated translation table to extract paraphrased sequences (pivot method). Each word is a source sentence with each grapheme being an element of the source sentence and each pronunciation is a target sentence with each phoneme forming an element of the target sentence.

Table 3.1 gives an overview of the data sets used with the number of entries (distinct pairs) and the average number of pronunciations per word in the three training conditions after preprocessing.

Table 3.1: *Training conditions*

Training set	Number of entries	Average number prons/word
tr_set	201,423	1.2
tr_set_l	162,974	1.0
tr_set_m	67,769	2.3

It can be seen in Table 3.1 that there are large differences between the three training

2. Although not publicly available, this dictionary is available by request. It has been used by numerous laboratories. SRI, Philips Aachen, ICSI and Cambridge University have reported improving the performance of their systems using this dictionary.

conditions. For tr_set_m the number of entries diminishes to one third of the original dictionary. However, the number of pronunciations per word almost doubles. In this case, the extra information given by the canonical pronunciations of words with only one pronunciation is lost, but we allow the system to change the frequency relationship between the phrases of the canonical pronunciations and the phrases found in pronunciation variants, and see how this influences the generation of pronunciation variants which is the main interest of the p2p conversion. In the second training condition, only the canonical pronunciation of each word is kept in the training data. This allows us to see if pronunciation variants can be generated even under limited training conditions. For example, this condition corresponds to generating variants from the output of a rule-based g2p system which, if originally developed for speech synthesis, may not model pronunciation variants or to enriching a dictionary with limited pronunciation variants.

3.4 Evaluation

In this Section the evaluation of the g2p and p2p conversion will be presented. A further evaluation of the generated pronunciations is effected adding these pronunciations to a state-of-the-art ASR system. These speech recognition results will be presented in the next Section (Section 3.5). In practice, it is much easier and less costly to assess the performance of a letter-to-sound module separately and in isolation from any other components (unit testing). This avoids the dependencies that can arise in end-to-end evaluation and is thus more suitable for an initail comparison of different techniques for g2p and p2p conversion (Jansche, 2003).

3.4.1 Definition of evaluation measures

In this study, precision and recall, first introduced in information retrieval (Van Rijsbergen, 1979), as well as phone error rate (PER) are used to evaluate the predictions of one or multiple pronunciations. We will first define these measure for the g2p conversion case. Word x_i of the test set ($i = 1..\mathtt{w}$) has j distinct pronunciations y_{ij} (y_i is a set with elements $y_{ij}, j = 1..d_i$). Moreover, our systems can generate one or more pronunciations $f(x_i)$ ($f(x_i)$ is also a set). Recall (R) is conventionally defined:

$$\mathrm{R} = \frac{1}{\mathtt{w}} \sum_{i=1}^{\mathtt{w}} \frac{|f(x_i) \cap y_i|}{|y_i|} \tag{3.5}$$

Precision (Pr) is defined analogously as the number of correct generated pronunciations divided by the total number of generated pronunciations. They are calculated on all references (canonical pronunciations and variants) to evaluate the g2p conversion, but also only on the variants in order to specifically evaluate their correctness.

We refer to the previous definitions as micro-recall and micro-precision respectively (Stroppa, 2005). If the examples are normalized by the number of expected pronuncia-

tions (correct pronunciations in the reference r_i), the total recall becomes:

$$r = \frac{\sum_{i=1}^{\mathtt{W}} |r_i||y_i|}{\sum_{i=1}^{\mathtt{W}} |y_i|}, \tag{3.6}$$

where $r_i = \frac{|f(x_i) \cap y_i|}{|y_i|}$ the recall of the word x_i. In this last case, the macro-recall is defined. Macro-precision is defined analogously. The macro-measures give more weight to the examples with multiple pronunciations, while the micro-measures consider all the examples equally weighted.

The PER is measured using the Levenshtein Distance (LD) between the generated pronunciations and the reference pronunciations and the mean value on the whole corpus is taken:

$$PER_{1-best} = \frac{\sum_{i=1}^{\mathtt{W}} PER(y_i, f(x_i))}{\mathtt{W}} \tag{3.7}$$

$$PER_{n-best} = \frac{\sum_{i=1}^{\mathtt{W}} \sum_{j=1}^{d_i} PER(y_{ij}, f(x_i))}{\sum_{i=1}^{\mathtt{W}} |y_i|}. \tag{3.8}$$

The idea is to compute the PER when only one pronunciation is generated by Moses (PER_{1-best}) and when an n-best list of pronunciations is generated by Moses (PER_{n-best}). The nominator of Equations 3.7 and 3.8 computes the PER of each word of the dictionary and sums the PERs of all the dictionary words. If only one pronunciation is generated by Moses (Equation 3.7), it is desired that this pronunciation is as close as possible to one of the reference pronunciations of the corresponding word. In order to compute the PER, the LD between all the reference pronunciations of the word and the one pronunciation generated by Moses is computed and the minimum of these LD scores is kept and normalized by the number of reference phonemes. Then, the denominator serves to take the mean value of the PERs of all reference words. If more pronunciations are generated by Moses (Equation 3.8), the LD between each pronunciation of the reference and each pronunciation generated by Moses is computed, and the minimum LD score for each reference pronunciation is kept. These minimum LD scores are again normalized and the PERs of the reference pronunciations are generated. In the n-best case, we have one minimum LD score for each reference pronunciation, and not one for each word as in the 1-best case. This is because we aim at generating as many correct reference pronunciation variants as possible and not just one correct pronunciation per word as in the 1-best case. Finally, the denominator serves to take the mean value of the PERs of all reference pronunciations.

The definitions are the same for the p2p conversion task, with the difference that in the upper equations x_i is the canonical pronunciation of the test set, for which a set y_i of one or more pronunciation variants is generated.

3.4.2 G2P conversion results

The Moses-g2p converter (M-g2p) and the pivot paraphrasing method (P) were tested for the entire training set (tr_set) and for the canonical pronunciation training conditions (tr_set_l). The PER on the test set for the training condition (tr_set_l) is presented in Table 3.2 for Moses-g2p and Pivot with LD2 pruning (P LD2) and in Table 3.3 for the training condition (tr_set). The PER for both training conditions is about 6% for the 1-best

Moses-g2p pronunciation, and about 1% if the 10-best pronunciations are considered. The similar results between the two training conditions are a good indication that our systems can generate pronunciations of good quality even under restricted variation conditions in training. Since the 1-best pronunciations generated by Moses-g2p are used as input to the pivot post-processing, the corresponding entry in the table is empty for Pivot. The string error rate (SER) is 25%.

Table 3.2: *PER on all references (canonical pron+variants) for Moses-g2p (M-g2p) and Pivot (P) for canonical pronunciation training*

Method	Measure	1-best	2-best	5-best	10-best
M-g2p	PER (%)	**6.22**	3.99	1.98	**1.26**
P LD2	PER (%)	-	6.17	5.16	3.52

Table 3.3: *PER on all references (canonical pron+variants) for Moses-g2p (M-g2p) and Pivot (P) for the entire training set condition*

Method	Measure	1-best	2-best	5-best	10-best
M-g2p	PER (%)	**6.13**	4.00	1.97	**1.17**
P LD2	PER (%)	-	6.00	4.47	3.52

Table 3.4 gives recall results compared to all references (top) and only variants (bottom) with both methods for canonical pronunciation training. Table 3.5 gives the corresponding results for training on the entire training set (keeping one or multiple pronunciations per word). Precision was also calculated, but only recall is presented because we consider it more important to cover possible pronunciations than to have too many, since other methods can be applied to reduce the overgeneration (alignment with audio, manual selection, use of pronunciation probabilities, etc). The best value that both precision and recall can obtain is 1.

It can be seen in Table 3.5 that, for the training on the entire training set, Moses-g2p outperforms the pivot-based method in terms of recall measured on all references (R-all ref) and on variants only (R-variants). The best result is a recall on all references of 0.94 when using the 10-best pronunciations generated by Moses-g2p.

In Table 3.4 the recall on all references (top) and only on variants (bottom) for canonical pronunciation training are shown. For the recall on variants, the results of pivot without LD pruning are presented (P) as well as with LD threshold 3 (P LD3) and LD threshold 2 (P LD2) to show the improvement obtained by the intermediate pruning steps (see Section 3.2.2 for a reminder on the pruning performed on the pivot result).

Comparing the recall on all references (R-all ref) in the two tables, a 3% absolute degradation can be seen in Table 3.4 for both methods. However, the variant-only recall degrades more severely. For the latter case (R-variants) pivot with LD2 or LD3 pruning outperforms Moses-g2p. It manages to generate more correct variants even when no variants are given in the training set (see lower part of Table 3.4). Pivot takes directly the variation patterns from the phrase table of Moses avoiding the overfitting effects of the

Table 3.4: *Recall on all references (canonical pron+variants) and only on variants for Moses-g2p (M-g2p) and Pivot (P) for canonical pronunciation training*

Method	Measure	1-best	2-best	5-best	10-best
M-g2p	R-all ref	0.68	0.79	0.88	**0.91**
P	R-all ref	-	0.72	0.78	0.82
P LD3	R-all ref	-	0.72	0.78	0.82
P LD2	R-all ref	-	0.72	0.78	0.83
M-g2p	R-variants	0.10	0.25	0.44	**0.55**
P	R-variants	-	0.19	0.32	0.44
P LD3	R-variants	-	0.35	0.49	0.60
P LD2	R-variants	-	0.36	0.50	**0.61**

Table 3.5: *Recall on all references (canonical prons+variants) and only on variants for Moses-g2p (M-g2p) and Pivot (P) for the entire training set condition*

Method	Measure	1-best	2-best	5-best	10-best
M-g2p	R-all ref	**0.68**	0.82	0.91	**0.94**
P LD2	R-all ref	-	0.74	0.80	0.84
M-g2p	R-variants	0.27	0.63	0.82	0.89
P LD2	R-variants	-	0.50	0.66	0.73

EM algorithm used by Moses for the construction of a generative model. Moreover, to reduce the overall complexity of decoding, the search space of Moses is typically pruned using simple heuristics and, as a consequence, the best hypothesis returned by the decoder is not always the one with the highest score.

It should be pointed out that the measures (recall and PER) on all references favors the Moses-based approach since the pivot-based approach aims to generate variants. This is why recall only on variants was also evaluated. However, while the pivot method gives better results than Moses-g2p to variants generation for the canonical pronunciation training condition, this is not the case when multiple pronunciations are used for training. Some additional analyses were carried out to investigate this further. When the pivot is used as a post-processing step to the Moses-g2p converter, its input is the output of Moses which has PER of 6%, low enough to be reliable, but the SER is 25% which can plausibly degrade the performance of pivot. To verify this hypothesis, the pivot method was applied to the correct canonical pronunciation of the test set and these results were compared to the previous results of 1-, 4- and 9-best variants generated by pivot as well as to the variants generated by Moses-g2p. In order to more clearly see the influence of variants generated by pivot, the 1-best pronunciation generated by Moses was not retained as had been done previously. This pronunciation was also removed from the n-best list generated by Moses-g2p in order to compare the two methods. Table 3.6 gives recall results computed on variants in the reference set. It can be seen that pivot, when applied to a correct input, not only outperforms itself applied to a "noisy" input, but also the Moses-g2p method. This is an important observation, as there are cases where the enrichment

Table 3.6: *Recall on variants only for generation of 1-, 4- and 9-best variants by Moses-g2p (M-g2p) and Pivot (P) for the entire training set condition*

Method	Measure	1-best	4-best	9-best
M-g2p	R	0.35	0.55	0.62
P LD2	R	0.23	0.39	0.46
P correct entry LD2	R	0.39	0.65	**0.75**

Table 3.7: *Macro-Recall on all references (canonical prons+variants) and only on variants for Moses-g2p (M-g2p) and Pivot (P) for both training conditions.*

Canonical pron training condition					
Method	Measure	1-best	2-best	5-best	10-best
M-g2p	mac-R-all ref	0.62	0.73	0.82	0.87
P LD2	mac-R-all ref	-	0.66	0.73	0.78
M-g2p	mac-R-variants	0.10	0.23	0.40	0.51
P LD2	mac-R-variants	-	0.33	0.46	0.56
Entire dict training condition					
Method	Measure	1-best	2-best	5-best	10-best
M-g2p	mac-R-all ref	0.62	0.78	0.89	0.93
P LD2	mac-R-all ref	-	0.69	0.77	0.82
M-g2p	mac-R-variants	0.25	0.56	0.79	0.88
P LD2	mac-R-variants	-	0.46	0.63	0.70

of a single pronunciation dictionary is desired, for example in a conversational speech transcription task.

All results presented in this section are calculated with the complete 45-phone set used in the LIMSI dictionary. However, some exchanges are less important than others. If some errors, such as the confusion between syllabic nasals and a schwa-nasal sequence, are not taken into account (a subset of those proposed in (Lee and Hon, 1989)), the overall recall improves by 1-2% absolute for both methods, and the PER is reduced by 0.1-0.2% for Moses-g2p and 0.3-0.4% for pivot.

Last but not not least, the reference dictionary is mostly manually constructed and certainly incomplete with respect to coverage of pronunciation variants particularly for uncommon words. The pronunciations of words of foreign origin (mostly proper names) may also be incomplete since their pronunciation depends highly on the speaker's knowledge of the language of origin. This means that some of the generated variants are likely to be correct (or plausible) even if they are not in the references used in the upper evaluation.

For the g2p conversion task, macro-recall gave similar results to the conventional recall. For the sake of completeness, these results are summarized in Table 3.7 for both training conditions.

3.4.3 P2P conversion results

The two systems, Moses as phoneme-to-phoneme converter (m_p2p) and the pivot paraphrasing method (p_p2p) were tested for the three training conditions presented in Section 3.3. The results using the two proposed evaluation metrics are shown in Tables 3.8 and 3.9 respectively. We only present recall measures in the tables because this is what is of most interest in the particular task. The best value that both precision and recall can obtain is 1. However, the best value of precision is often further limited depending upon the number of elements of the n-best list and the overgeneration that cannot be avoided. The n-best list is limited to 10 because preliminary studies showed that larger n only slightly improves recall while severely degrades precision. There is quite a bit of overgeneration, since in the 19k pronunciation-pronunciation pairs of the test set there are only 4k pairs with pronunciation variants. This could not be avoided with a random selection of the test set from the original dictionary where only 18% of words have variants as already stated. However, there is the possibility that some of the generated variants which are not in the reference (and therefore counted as errors) could be considered acceptable by a human judge.

As can be expected, for both methods the number of correctly generated variants increases with the size of the n-best list. This is normal not only because the number of hypothesis increases with the size of the n-best list, but also because there are canonical pronunciations in the test set that have more than one variant (approximately 1/6 of the part of the test set with multiple pronunciations) which cannot be captured when an insufficient number of pronunciations is generated.

It is also interesting to compare the results of the first and the third training conditions (entire training set vs. keeping only entries with multiple variants) for the two methods. In the second case, the amount of training data is only one third of the original training set. However, the results are more or less the same for both training conditions. This may be because the information that the model is using to learn how to generate variants is mostly captured by the multiple pronunciations in the training set and less by the fewer variations observed in the canonical pronunciations of one-pronunciation words. What the model is learning in this case is focused on the relationship between the canonical pronunciation and other variants, and therefore has effectively more relevant information and it does not get watered down by the self-production. This may compensate for the

Table 3.8: *Results using Moses as phoneme-to-phoneme converter for the 3 training conditions*

Training set	Measure	1-best	5-best	10-best
tr_set	Micro-recall	0.20	0.75	0.83
	Macro-recall	0.19	0.73	0.81
tr_set_l	Micro-recall	–	0	0
	Macro-recall	–	0	0
tr_set_m	Micro-recall	0.21	0.75	0.80
	Macro-recall	0.19	0.74	0.80

Table 3.9: *Results using the pivot paraphrasing method for the 3 training conditions*

Training set	Measure	1-best	5-best	10-best
tr_set	Micro-recall	0.29	0.60	0.70
	Macro-recall	0.26	0.56	0.66
tr_set_l	Micro-recall	0.09	0.26	0.38
	Macro-recall	0.09	0.24	0.35
tr_set_m	Micro-recall	0.25	0.56	0.70
	Macro-recall	0.22	0.53	0.66

reduced amount of data.

A comparative analysis of the two methods can also be made. In the first (entire training set) and in the third (only entries with variants) training conditions, using Moses as a p2p converter gives better results in terms of the generation of pronunciation variants for both micro and macro measures when the 5-best and the 10-best variants are kept. However, when only the 1-best generated pronunciation is kept, the pivot method gives better results. This is due to the generation of canonical pronunciations by Moses when used as p2p converter, which are subsequently removed from the results because they already exist in the input. The number of variants generated by Moses-p2p when only the 1-best is kept are quite limited. This is why, while the recall is lower than that of the pivot method, the precision is higher.

It can be seen that the results change when the training set is limited to the canonical pronunciations only (the second condition). In this case the pivot method manages to produce some results, while for Moses the model fails to generate any variants (this is why the corresponding columns are left empty in Table 3.8) and all the variants that are in the 5-best or the 10-best lists are false. These results warrant a bit more discussion. The results are promising for the pivot method, because even when the training dictionary has few or no pronunciation variants, the pivot method can still be used to generate some alternative pronunciations. This can be explained by the fact that the pivot method uses also the graphemic information. Even if no variants are included in the training set, it can still find graphemic sequences of words that correspond to different phonemic sequences and consider these phonemic sequences as possible modifications of pronunciations. For example, in the training set the word "autor**out**e" is pronounced /ctor**ut/** and the word "sh**out**ing" is pronounced /S**Wt**IG/. These words have the graphemic sequence "**out**" in common which can be used as a pivot between the phonemic sequences **/ut/** and **/Wt/**. These phonemic sequences become a paraphrased pair that generates correctly the variants /r**Wt**s/ and /r**ut**s/ of the word "r**out**es" found in the test set.

This also illustrates the difficulty of generating pronunciations in English, because the correspondence between orthographic forms and canonical pronunciations does not follow strict rules which would prevent the pivot method from finding modified phonemic sequences corresponding to the same graphemic sequence. This is not the case when Moses is used as phoneme-to-phoneme converter. When no variants are given to the system, it does not have any additional information in order to be trained for the task of generating multiple pronunciations. It is like trying to train an SMT system without a

target language. It can just learn to align the phonemic sequences with themselves, which is not applicable to the generation of variants. In this case, is it wrong to use Moses for this task as it is obvious that it has nothing to learn from the training data.

3.5 Speech recognition experiments

The pronunciations generated by the Moses-g2p were further tested in two speech recognition experiments. In the first, automatically generated variants are added to a single-pronunciation dictionary, and in the second we simulate adding pronunciations for OOV words. To our knowledge it is the first time they are tested in a state-of-the-art ASR for English broadcast data.

The speech transcription system uses the same basic modeling and decoding strategy as in the LIMSI English broadcast news system (Gauvain et al., 2002). The speech recognizer makes use of continuous density HMMs with Gaussian mixture for acoustic modeling and 4-gram statistics estimated on large text corpora. The acoustic models (AMs) are gender-dependent, word position dependent, speaker-adapted, and Maximum Likelihood trained on about 500 hours of audio data. They cover about 30k phone contexts with 11,600 tied states. N-gram LMs were trained on a corpus of 1.2 billion words of texts from various LDC corpora (English Gigaword, BN transcriptions, commercial transcripts), news articles downloaded from the web, and assorted audio transcriptions. The recognition word list contains 78k words, selected by interpolation of unigram LMs trained on different text subsets as to minimize the OOV rate on a set of independent development texts. The transcription system has two main components, an audio partitioner based on an audio stream mixture model (Gauvain et al., 1998) and a word recognizer. For each speech segment, the word recognizer determines the sequence of words, associating start and end times and an optional confidence measure with each word. Word recognition was performed in a single decoding pass with decoding time being roughly the duration of the audio signal, generating a word lattice with cross-word, position-dependent AMs, followed by consensus decoding (Mangu et al., 1999) with a 4-gram LM. Unsupervised AM adaptation is performed for each segment cluster using the CMLLR and MLLR techniques prior to decoding.

The Quaero (www.quaero.org) 2010 development data were used in these experiments. This 3.5 hour data set contains 9 audio files recorded in May 2010, covering a range of styles, from broadcast news (BN) to talk shows. Roughly 50% of the data can be classed as BN and 50% broadcast conversation (BC). These data are considerably more difficult than pure BN data. The overall word error rate (WER) with the original recognition dictionary is 30%, but the individual shows vary from 20% to over 40%. These are competitive WERs on these data. The WERs for the individual shows are shown in Table 3.10. The results with baseline system, which has an average of 1.2 prons per word, can be compared to that using a single pronunciation dictionary (the most frequent based on automatic alignments of the audio training data).

In Table 3.11, the n-best pronunciations (1-, 2- and 5-best) generated by the Moses-based system under the two training conditions, are added to the canonical pronunciation of the original recognition dictionary (Baseline longest). The results show that using

Table 3.10: *Quaero 2010 development data set composition and baseline word error rates (%) with the original dictionary and a single pronunciation one.*

Show	Duration	Baseline	1 pron
20100305_BBC_WH	16 mn	33.1	36.6
20100305_BBC_PHONEIN	35 mn	43.1	44.3
20100305_NEWSPOD	37 mn	24.4	27.3
20100305_BBC_MAYO	15 mn	33.5	38.1
20100308_NEWSPOD	32 mn	23.6	26.9
20100308_NAKED_SCIENTISTS	18 mn	23.1	28.3
20100310_BBC_MIDWEEK	15 mn	30.7	33.7
20100310_BBC_MEDIASHOW	26 mn	30.3	32.9
20100311_CNN_NEWS	27 mn	22.0	25.3

only the longest pronunciation (line "Baseline longest": WER 41,6%) results in a large increase in WER compared to using the original recognition dictionary which includes common variants (WER 30%). Adding pronunciations improves over the baseline longest dictionary, up until the 5-best pronunciations. The pronunciations trained using the entire training set (one or multiple pronunciations per word) improve more the WER compared with the pronunciations trained with the canonical pronunciation dictionary. This is because the former are trained to better model the variants which correspond to the reduced forms, closer to the spoken language most of the times.

In Table 3.12, the same pronunciations (M1, M2, M5) are added to the most frequent pronunciation of the recognition dictionary (Baseline most frequent). The most frequent pronunciation baseline dictionary has a WER closer to the baseline of the original multiple pronunciation dictionary. In this case adding one pronunciation (for both training conditions) improves the performance of the ASR system, but adding more pronunciations degrades it.

In both Tables 3.11 and 3.12 it can be observed that the best results are reported when only one variant is added to the lexicon (column "M1"). This suggests that adding more variants actually harms the ASR performance, even if these variants actually improve the recall score in the g2p conversion evaluation (see Section 3.4). This observation introduces the confusability problem which we will try to better understand and moderate in the following chapters of this thesis.

Although the quality of the pronunciations trained on a multiple pronunciation dictionary is higher, measured with recall on all references and on variants, they are submitted

Table 3.11: *WER(%) adding Moses nbest-lists (M1, M2,M5) to the longest pronunciation baseline*

Training condition	M1	M2	M5
Canonical pronunciation	38.2	38.4	40.8
Entire training set	37.9	38.2	39.1
Baseline longest	41.6		

Table 3.12: *WER(%) adding Moses nbest-lists (M1, M2,M5) to the most frequent pronunciation baseline*

Training condition	M1	M2	M5
Canonical pronunciation	32.0	33.4	37.3
Entire training set	32.0	34.5	38.9
Baseline most frequent	32.9		

Table 3.13: *WER(%) adding Moses nbest-lists (M1, M2,M5) to the original LIMSI dictionary*

Training condition	M1	M2	M5
Entire training set	31.1	33.8	38.9
Original LIMSI dictionary	30.0		

to the same confusability effects. What is more, when adding two or five pronunciations to the most frequent baseline, the system with pronunciations trained on a single canonical pronunciation presents lower WERs. An explanation could be that the pronunciations trained under multiple pronunciations can better represent reduced forms and, thus, are closer to the most frequent baseline and easier to be confused. An example of the introduced confusability is that of the multiple pronunciation training outputs for the entry of the dictionary "they're". There are the pronunciations */DeX/* and */Der/* when the 2-best list is kept. The latter pronunciation (*/Der/*) is not generated under the canonical pronunciation training. */Der/* is an homophone with the pronunciations of the word "their". Such frequent cases can be responsible for the degradation of the ASR system with pronunciations trained on the multiple pronunciation dictionary, when many alternatives are added.

We also tried to add the same pronunciations (M1, M2, M5) to the original multiple pronunciation LIMSI dictionary, but no improvement was observed even when just one additional pronunciation was included in the original recognition dictionary (Table 3.13).

We then simulated the generation of pronunciations for OOVs. Starting with the full dictionary with variants, the pronunciations for the 20% least frequent words in the test data (about 7% of dictionary) were replaced with automatically generated pronunciations using the Moses g2p system. For comparison, the g2p system l2s (Wasser, 1985), obtained from the Cambridge University ftp server and slightly modified to use the phone set of the LIMSI ASR system, was used to generate pronunciations for the missing words. (Lamel and Adda, 1996) reports that this system provided consistent pronunciations and gave satisfactory recognition results. The results in Table 3.14 show that the dictionary with Moses-based pronunciations for the OOVs outperforms those of the l2s system, even with 10 pronunciations, even though these many alternatives can be expected to cause confusions.

Nevertheless, in neither case was the performance of the original multiple pronunciation dictionary achieved. This dictionary is a difficult baseline because it is mostly manually constructed and well-suited to the needs of an ASR system.

Table 3.14: *WER(%) generating prons for OOVs using l2s and Moses nbest-lists (M1, M2, M5, M10).*

l2s	M1	M2	M5	M10
37.8	31.3	31.2	32.0	33.2

3.6 Conclusion

This chapter has reported on applying two data-based approaches inspired from research in statistical machine translation to the problem of generating pronunciation variants, both methods fully automatic and language independent. The main contribution of this work is that we propose a novel pivot-based method and compare this approach with directly taking the n-best lists from a Moses SMT system. A general comment can be that the n-best lists of Moses have a higher recall when a comparison is made to all reference pronunciations in the test set. However, the pivot-based method generates more correct variants. This is an advantage of the pivot method that could be useful in certain cases, for example to generate variants from the output of a rule-based g2p system which, if originally developed for speech synthesis, may not model pronunciation variants or to enrich a dictionary with limited pronunciation variants.

When used for p2p conversion, the approaches differ in the way that information about pronunciation variation is obtained. The approach using Moses as phoneme-to-phoneme converter takes into account only the information provided by the phonemic transcriptions. The pivot method uses information from both the phonemic and the orthographic transcriptions.When the full dictionary (that contains words with one or more pronunciations) is used for training, the Moses-based method gives better results than the pivot-based one. This is also the case when training is carried out only on entries with multiple pronunciations. However, when the training dictionary does not contain any pronunciation variants, the Moses-based method cannot be used, while pivot can still learn to generate variants. This is an advantage of the pivot method, and could be useful for languages without well-developed multiple-pronunciation dictionaries. This arises from the use of information provided by the orthographic transcription by the pivot method.

Another important outstanding issue concerns the proper way to evaluate the ability of a system to generate pronunciation variants. In this work, recall and PER have been used, however other measures could also be applied. For example, in the case of phoneme accuracy it may be appropriate to have phone-class dependent penalties with certain confusions being more important than others. Moreover, some pruning could be applied on the generated variants in order to imporve precision. One direction to explore is using audio data to remove pronunciations, however this can only apply to words found in the audio data. The reader is referred to (Yvon et al., 1998) for a discussion on open questions concerning the evaluation of g2p systems.

The pronunciations generated by Moses were also used to carry out some tests in a state-of-the-art ASR system. These experiments show that the added pronunciations are of good quality even when trained under limited variation conditions and can improve the single pronunciation baselines. When the generation of pronunciations for OOVs is

simulated, they outperform a Cambridge l2s system and they give results close to the baseline. Our point is not, however, to focus on the improvement of the performance of an ASR system, but to propose data-driven approaches for variant generation that better model variability in spoken language which can be useful in several applications.

Chapter 4

Measuring the pronunciation confusability in ASR decoding

In the previous chapter, the focus was on g2p and p2p conversion and the generated pronunciations and variants were added to a state-of-the-art word recognizer. A basic observation of this work, and at the same time a well-known problem of pronunciation modeling, was the confusability introduced to the system once alternative pronunciations are added to the recognition dictionary. This confusability, as already mentioned, can severely degrade the ASR performance, especially if the weights of the pronunciation variants are not suitably trained. This is an extra amount of confusability that is added to the "basic" confusability of the pronunciation dictionary. The "basic" confusability is a compromise between the size of the dictionary and the OOV rate, meaning that a smaller in size dictionary (with less words) will have less homophones and thus less confusability, but it will present a higher OOV rate. Therefore, it can result in a worse ASR performance, this time not because of confusability issues, but because of words missing in the dictionary. How to measure and reduce this "basic" as well as the extra confusability when variants are added is an open problem, because homophones are an inherent phenomenon of speech and we cannot just discard them without negatively affecting the ASR performance.

In this chapter, we focus on understanding more about the confusability inherent to the pronunciation dictionary. In particular, we define a measure aimed at assessing how well a pronunciation model will function when used as a component of a speech recognition system (Section 4.2). This measure, *pronunciation entropy*, fuses information from both the pronunciation model and the language model and measures the uncertainty introduced by these components in the system. We then show how to compute this score by effectively composing the output of a phoneme recognizer with a pronunciation dictionary and a language model, and investigate its role as predictor of pronunciation model performance. In Section 4.4 we present results of this measure for different dictionaries with and without pronunciation variants and counts.

4.1 Introduction

A lot of work has been carried out on the generation of pronunciations and pronunciation variants independently of the speech (g2p conversion, p2p conversion) or in a task specific framework using surface pronunciations generated from a phoneme recognizer or including acoustic and language model information. However, despite the inclusion of the acoustic model (Holter and Svendsen, 1999), (Fosler-Lussier and Williams, 1999) and language model influences (Kessens et al., 1999) into the pronunciation modeling process, most works lack a sense of how the added alternative pronunciations will affect the overall decoding process. Confusability is not limited to the phonemic proximity between homophones but includes also ambiguities provoked by the segmentation of the phonemes to words, and in the same time is influenced by the LM scores. This is novel compared to previous works, such as the work of (Fosler-Lussier, 1999), where the entropy is defined as a local measure computed on predicted units (phones/syllables/words).

Figure 4.1 gives an example of the ambiguity that can be created during segmentation into words of a phoneme sequence. For the sake of simplicity, let's consider that a simple phoneme sequence is generated by a phoneme recognizer. Usually a phoneme lattice is generated which can make the segmentation ambiguities a lot larger. Thus, a simple phoneme sequence when composed with the pronunciation dictionary is recognized as two phrases of different length and word content. However, some of the extra confusability introduced by the pronunciation model may be compensated by the LM, whose scores may help the decoder to recognize the right sentence in the end. Thus, a method for quantifying the confusion in a combined acoustic-lexical system is needed.

A confusability measure traditionally used to measure the uncertainty residual to a system is the entropy. Specifically in an ASR system, lexical entropy measures the confusability when a certain LM is used. In some previous works, the scores of the acoustic and pronunciation models are used in addition to the LM scores in order to measure lexical entropy (Printz and Olsen, 2000). In (Wolff et al., 2002), the authors consider the entropy of the variant distribution as a measure of the pronunciation confusability, but they do not take into account the language model, and in (Fosler-Lussier, 1999) only a prior distribution on pronunciations is used for the entropy calculation. Our aim is to integrate pronunciation model and language model information into a single framework for describing the confusability. Especially incorporating language model information would provide a more accurate reflection of the decoding process, and hence a more accurate picture of the possible lexical/acoustic confusions (Fosler-Lussier et al., 2002). The idea is then to introduce a measure inspired by the formulation proposed in (Printz and Olsen, 2000) but in a somewhat reverse fashion. Instead of measuring the "true" disambiguation capacity of the LM by taking acoustic similarities into account, we aim at measuring the actual confusability introduced in the system by the pronunciation model, taking also into account the LM. We call this measure *pronunciation entropy*.

To compute this measure, we will decompose the decoding process in two separate parts: the acoustic decoding on the one hand, the linguistic decoding on the other hand. Given an input signal, a phoneme recognizer is first used to obtain a sequence of phonemes; the rest of the decoding process is realized using a set of Finite State Machines

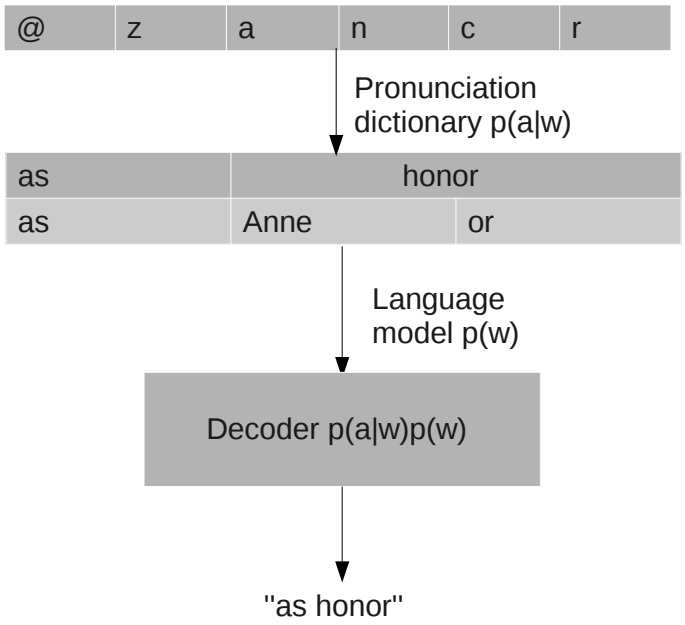

Figure 4.1: *Example of ambiguity during word segmentation in ASR decoding*

(FSMs) modeling the various linguistic resources involved in the process. This is not optimal from an ASR point of view, but allows us to carry a further study on the components of the decoder. Doing so allows us to measure the confusability incurred by the acoustic decoder for fixed linguistic models; or, conversely, to assess the impact of adding more pronunciations, for fixed acoustic and language models. This latter scenario is especially appealing, as these measurements do not require to redecode the speech signal: it thus become possible to try to iteratively optimize the pronunciation lexicon at a moderate computational cost. Experiments are carried out to measure the confusability introduced by single and multiple pronunciation dictionaries in an ASR system of continuous speech, using the newly introduced pronunciation entropy.

4.2 A new confusability measure

4.2.1 ASR decoding with FSTs

The recognition process can be modeled with a sequence of weighted finite-state transducers (WFSTs) (Pereira and Riley, 1996). An abstract representation of the Viterbi decoding process of the present work can be given as:

$$\hat{W} = bestpath(A \circ P \circ L \circ G), \tag{4.1}$$

where $\hat{W}$ is the sequence of words corresponding to the best recognition hypothesis. A is the phoneme hypothesis lattice generated by the phoneme recognizer, P is an FST that contains a mapping from phonemes to the phonemic lexical representation of each word, L is the pronunciation model FST, containing a mapping from each phonemic lexical representation to the corresponding word, G is the language model finite state automaton (FSA), which contains n-gram statistics, and $\circ$ is the composition operator. Constraining the model by the pronunciation and the language models means that only words that are part of complete paths in the decoding will be counted as confusions.

The same model of decoding as a series of formal compositions will be used for the work presented in Chapter 5.

4.2.2 Decomposing the acoustic and linguistic modeling

In a first place, a phoneme recognizer generates the phoneme hypothesis lattice A from the speech signal. These phonemes are the input in the following process of consecutive compositions. The phoneme lattices are generated by the ASR system without taking into account the pronunciation nor the language model during decoding. The aim is to decompose the decoding parts in order to better evaluate the influence of the pronunciation model in the decoding process. The acoustic scores are considered stable and independent of the linguistic (pronunciation and language) confusability and thus are omitted. Later, however, to have a final score as close as possible to the real decoding score, the acoustic scores will be kept and, in addition, a weight scaling (LMscale) will be applied on the language model. No time information is kept in the lattice. The pronunciation model will automatically segment the phoneme sequences in pronunciations, and consequently in words. All results will be presented in Section 4.4.

Then, the FST P is constructed, which represents the pronunciations of our lexicon (see Section 4.3). Each path of P has as input a sequence of phonemes and returns the corresponding phonemic lexical representation (pronunciation). P is composed with each phoneme lattice. In order to account for insertions of phonemes between valid pronunciations of our lexicon, the topology of P is slightly expanded. This expansion simulates a simple error recovery strategy consisting in deleting superfluous phonemes in a left to right fashion. Fig. 4.2 illustrates this expansion on a simple example, with the use of failure transitions implemented with the so-called *phi-matchers* and *rho-matchers* (see Section 2.3). Each state in P corresponds to the prefix of an actual pronunciation: whenever we reach a state from which no continuation is possible, a phi-transition allows us to reach the state corresponding to a trimmed prefix, from which the first phoneme has been deleted. This simple error recovery strategy is applied recursively. A rho-loop is finally used in the initial state, which is also the final state, in case the first or last phonemes of a sequence do not permit to complete a known pronunciation. Assume, for instance, that we reach state 2 in P (see Fig. 4.2), and that the following symbol is 'c', for which no transition exists. The phi-transition will then enable to move to state 3, and to continue the prefix 'bc'.

Next, the FST L representing the pronunciation dictionary with pronunciations as inputs and words as outputs is constructed. Its weights are the conditional probabilities

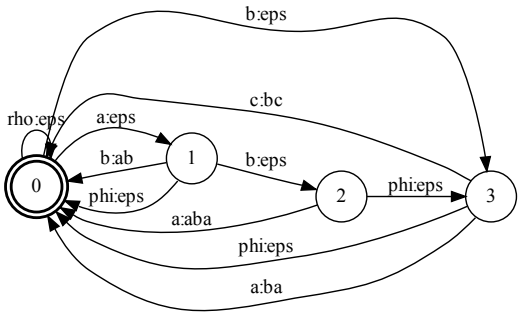

Figure 4.2: *Expansion of the topology of the P FST with phi and rho matchers that consume the phonemes inserted between valid pronunciations*

of a pronunciation given a word. When no pronunciation probabilities are available, a uniform distribution over the probabilities of pronunciations of each word is applied. This FST is composed with each phonemes-pronunciations FST ($A \circ P$) resulting from the previous composition.

A final composition is made with the FSA G that models a backoff language model, with word probabilities as weights. G is constructed as described in (Riccardi et al., 1996; Allauzen et al., 2003). This results in FSTs with phonemes as input and words as output, which are projected to the output and determinized. Then, the arc weights of each FST are normalized per state, i.e. scaled such that the probability of arcs leading out of a state (plus the probability of being final) sums to 1 for each state. A general weight-pushing algorithm in the log semiring (Mohri, 1997) is applied for the normalization and the weights in the new stochastic FSA are converted to the desired posterior probabilities given the pronunciations (see Section 2.3.5). What is calculated is the conditional probability $p(\mathtt{w} \mid a)$ of all the word sequences that can be transcribed as a and, thus, are competitors:

$$p(\mathtt{w} \mid a) = \frac{p(a \mid \mathtt{w})p(\mathtt{w})}{\sum_{\mathtt{w}' \in W} p(a \mid \mathtt{w}')p(\mathtt{w}')}. \tag{4.2}$$

4.2.3 Definition of pronunciation entropy

We define as *pronunciation entropy* a measure of the actual confusability introduced to an ASR system by the pronunciation model. It is defined in a sentence level to take into account the ambiguities of the word segmentations performed by the recognition dictionary. It integrates the LM scores and, plausibly the acoustic scores, to provide a global reflection on the decoding process.

In order to have a measure of the confusability of the pronunciation lexicon, the entropy of the posterior probability $p(\mathtt{w} \mid a)$ that combines the pronunciation model and the language model information is computed. As described in Section 2.3.6, computing the

shortest distance on the expectation semiring can result in the desired entropy. However, the expectation semiring must have components in the real semiring in order to compute the entropy correctly, but even real numbers of double precision are not stable enough for large lattices. Thus, an expectation semiring with components in the log semiring is needed. That is why we define a new semiring, the log-expectation (or log-entropy) semiring, changing the $\oplus$ and $\otimes$ operators as well as the identities of the semiring. In this new semiring $(\mathbb{K}, \oplus, \otimes, (\infty, 0), (0, 0))$, $\mathbb{K}$ denotes $(\mathbb{R} \cup [-\infty, \infty]) \times (\mathbb{R} \cup [-\infty, \infty])$ and the operations $\oplus$ and $\otimes$ on weight pairs (a_1, b_1) and (a_2, b_2) in $\mathbb{K}$, are defined as:

$$(a_1, b_1) \oplus (a_2, b_2) = (-log(exp(-a_1) + exp(-a_2)), b_1 + b_2) \tag{4.3}$$

$$(a_1, b_1) \otimes (a_2, b_2) = (a_1 + a_2, exp(-a_1)b_2 + exp(-a_2)b_1) \tag{4.4}$$

When working on the log-entropy semiring, each negative log arc weight w is replaced by the new weight $(w, w * exp(-w))$. Then, the shortest distance from the initial to the final state is computed as explained in Section 2.3.6. Some experiments were performed on small exemplar lattices with real arc weights and the entropy was calculated directly with the entropy semiring, already defined in OpenFst. However, for larger lattices the use of the log and the log-entropy semirings was required for numerical stability reasons.

4.3 Phoneme recognition

The phoneme recognizer used in these experiments is built using acoustic models that are tied-state, left-to-right 3-state HMMs with Gaussian mixture observation densities (typically 32 components). The acoustic models are word position independent, gender-dependent, and Maximum Likelihood trained on about 500 hours of audio data. It should be noted that the phone contexts cover both within word and cross word triphones, but are independent of the phone position in the word. This is actually different from the acoustic models that were used in the word recognition experiments reported in Chapter 3, where word position is explicitly taken into account with different phonemes expressing different word positions (word initial/final/internal triphones). The states are tied by means of a decision tree to reduce the model size and to increase triphone coverage. State-tying constructs one tree for each state of each phone so as to maximize the likelihood of the training data using single Gaussian state models, penalized by the number of tied-states. Silence is modeled by a single state with 1,024 Gaussians. The acoustic models cover about 30k phone contexts with a total of 11,500 tied states.

Unsupervised acoustic model adaptation is performed for each segment cluster using the CMLLR and MLLR techniques prior to decoding. Concerning the cepstral features, the 39 cepstral parameters are derived from a Mel frequency spectrum, with cepstral mean removal and variance normalization carried out on a segment-cluster basis, resulting in a zero mean and unity variance for each cepstral coefficient. A 3-dimensional pitch feature vector (pitch, Δ and $\Delta\Delta$ pitch) is combined with the cepstral features, resulting in a total of 42 (plpf0) or 81 parameters (mlpplpf0).

It suffices to say that the phone labels that are produced at that stage are deterministically mapped to the corresponding phonemes, which constitute the actual labels in the phoneme lattice. The recognition dictionary is a simple lexicon made up of the same list of phonemes used to represent pronunciations in the word lexicon (see Section 3.3). A 3-gram phoneme-based language model was also constructed to respond to the demands of the system for language modeling. A phoneme lattice is thus generated after a single decoding pass. The lattices are pruned so as to limit them to a reasonable size. To circumvent the fact that a lattice does not always finish with an end-of-sentence symbol, which can be the case because of the use of a time-based segmentation, an end-of-sentence symbol is added before the final state of each lattice.

The FST approach described in Section 4.2.1 is applied for word decoding. A 4-gram word LM is used, trained on a corpus of 1.2 billion words of texts from various LDC corpora (English Gigaword, Broadcast News (BN) transcriptions, commercial transcripts), news articles downloaded from the web, and assorted audio transcriptions. The recognition word list contains 78k words, selected by interpolation of unigram LMs trained on different text subsets as to minimize the out-of-vocabulary (OOV) rate on set of development texts. The recognition dictionary used as a baseline is the LIMSI American English recognition dictionary with 78k word entries with 1.2 pronunciations per word. The pronunciations are represented using a set of 45 phonemes (Lamel and Adda, 1996). This dictionary is constructed with extensive manual supervision to be well-suited to the needs of an ASR system. Other dictionaries with and without counts and variants were also tested, as described in the next section. These are the same lexicon and LM as the ones used in Section 3.5.

A part of the Quaero (www.quaero.org) 2010 development data was used in the recognition experiments. This data set covers a range of styles, from broadcast news (BN) to talk shows. Roughly 50% of the data can be classed as BN and 50% broadcast conversation (BC). These data are considerably more difficult than pure BN data. These are the same data used for word recognition when testing the g2p and p2p converters in Chapter 3. We chose to use the same data as well as the same lexicon and LM to be consistent in the experimental protocols used throughout the thesis and to allow for a further comparison and analysis of the results. The part of the Quaero data that was used resulted in 285 lattices generated by the phoneme recognizer. This is a sufficient number of lattices to have results that can be generalized. The FST-based decoding is applied to these lattices.

Table 4.1 summarizes some of the characteristics of the lattices and FSTs used in the composition process: the average number of states and arcs of the lattices A and of the phonemes-to-pronunciations FST P, of the pronunciations-to-words FST L and of the 4-gram word LM FSA G. Their size indicates that we are working in a real ASR framework with FSTs of large size. Thus, there is an important computational gain by the fact that the FST approach permits to change a part of the decoding without repeating the whole process.

Table 4.1: *Average number of states and arcs in the lattices and FSTs*

Num.	A	P	L	G
States	442	180,833	2	83,367,599
Arcs	650	503,234	171,272	200,322,203

4.4 Pronunciation entropy results

The presented pronunciation entropy is an average of the entropy calculated on the FSAs of the word sequences generated after the application of the FST decoding on the output lattices of the phoneme recognizer. The pronunciation entropy is calculated for the baseline dictionary with and without frequency of occurrence counts. When the frequency counts are not taken into account, a uniform probability distribution is assigned to the pronunciations of each word ("Baseline with uniform probabilities" in Table 4.2). The pronunciation entropy is also calculated for the "longest" baseline (keeping only the longest pronunciation per word in the original recognition dictionary) and the "most frequent" baseline (keeping only the most frequent pronunciation per word in the original recognition dictionary based on counts collected on the training data). The highest order language model (LM) used in the decoding of the word recognition experiments is a 4-gram.

In Table 4.2, the pronunciation entropy is presented when 2-, 3- and 4-gram LMs are used for the FST-based decoding. As expected, as the order of the LM diminishes, the entropy increases. The results when the order of the LM diminishes warrant some more thoughts. The difference in entropy even between the use of 4-gram and 2-gram is smaller than expected. The decoding is actually restricted by the given lexicon, that does not permit pronunciations, and thus words, to be correctly recognized if there is an error in the phoneme sequence. By manual observation of the best word hypotheses and their comparison with the corresponding references, it was thus noticed that only a small number of long n-grams are correctly recognized and, consequently, the impact of using a longer n-gram is relatively limited. However, it can also be an indication that there are confusions that do not get compensated even when an LM with larger context is used.

Table 4.2: *Pronunciation entropy with different dictionary baselines*

4g LM	**3g LM**	**2g LM**
Baseline with uniform probabilities		
4.095	4.101	4.284
Baseline with counts		
4.071	4.087	4.298
Longest Baseline		
3.673	3.679	3.877
Most frequent Baseline		
4.092	4.122	4.355

It is worth staying a bit longer in Table 4.2 to compare the pronunciation entropy of the baselines which contain one or more pronunciations per word (upper part of the table) and the single-pronunciation baselines (lower part of the table). The entropy is lower in the single pronunciation baselines, and its lowest score is observed when the "longest" baseline is used. The fact that its entropy is lower even when compared with the "most frequent" baseline, which is also a single-pronunciation baseline, may be explained by the fact that the most frequent pronunciations represent better the spoken terms that can be often easily confused. Especially in spontaneous speech, some function words are often pronounced similar to other function words and may not be easily distinguished by the LM. This is particularly a problem for frequent words that are easy to insert, delete or substitute.

A final observation from Table 4.2, comparing "Baseline with uniform probabilities" and "Baseline with counts", can be that pronunciations with counts do reduce confusability. It is normal not to see a lot of changes because in any case the majority of words has only one pronunciation and thus probability 1 which do not change when counts are added. In addition, counts are not available for all words, but only for those observed in the training data. When no counts are available, uniform probabilities are applied. Thus, finally there are no great differences between the dictionary with counts and the dictionary without counts. Even so, this result enforces the idea that if we want to moderate confusability we should add adequate weights to the pronunciation variants and not just remove them, which could be detrimental to the recognition performance.

The increase in entropy is much greater when more pronunciations are added in the dictionary as can be seen in Tables 4.3 and 4.4. The n-best pronunciations are added in the "longest" and the "most frequent" baselines. The M1, M2 and M5 in these tables correspond to the 1-, 2- and 5-best pronunciations generated automatically using Moses (Koehn et al., 2007) as a g2p converter, being trained on the baseline dictionary (with 1.2 pronunciations per word). Moses has been successfully used as a g2p converter for several languages, and for English it gives state-of-the-art results (Karanasou and Lamel, 2011) (see Chapter 3) . The results in Tables 4.3 and 4.4 are computed with the 4-gram LM.

Table 4.3: *Pronunciation entropy with the 4-gram LM after adding n-best pronunciations, produced by a Moses-based g2p converter, to the "longest" baseline*

Training condition	M1	M2	M5
Multiple pronunciations	4.234	5.252	6.228
Longest Baseline	3.673		

Table 4.4: *Pronunciation entropy with the 4-gram LM after adding Moses' n-best pronunciations to the "most frequent" baseline*

Training condition	M1	M2	M5
Multiple pronunciations	4.537	5.255	5.937
Most frequent Baseline	4.092		

Table 4.5: *WER(%) adding Moses nbest-lists (M1, M2,M5) to the single pronunciation dictionaries*

Dictionary	Baseline	M1	M2	M5
Longest pron	41.6	37.9	38.2	39.1
Most frequent pron	32.9	32.0	34.5	38.9

These results suggest that there can be a large influence of the pronunciation dictionary in the confusability of an ASR system, not sufficiently compensated by the language model. However, when adding as many alternative pronunciations some non-uniform probabilities should be used to moderate confusability. If not, the uniform probability contributed to each variant of a word with multiple pronunciations is lower. Thus, for highly probable words, since the system will have the tendency to choose them, the confusability will increase. But if the pronunciation probabilities are also taken into account, this confusability can be moderated, because a pronunciation of a word with lower probability and lower confusability (higher pronunciation probability) can be preferred over a pronunciation of a word with higher probability but lower pronunciation probability.

Observing the table 4.5 there is not a clear correlation shown between the confusability measured by the pronunciation entropy and the word error rate of the system. For both single pronunciation dictionaries ("Baseline longest" and "Baseline most frequent") the baseline WER is worse compared to some (or all) of the WER values once the Moses' n-best pronunciations are added to the system. However, the same baseline dictionaries present the lowest (best) pronunciation entropies. The only correlation that can be observed is if we limit the comparison to the WER values once the Moses' n-best pronunciations are added (columns "M1", "M2" and "M5"). The WER degrades as more variants are added, which correlates with the augmentation of the pronunciation entropy for the same cases in Tables 4.3 and 4.4.

4.5 Conclusion

A new measure of the confusability of the pronunciation model in a sentential basis during the decoding phase of an ASR system was presented. Results were reported using baseline dictionaries with one or more pronunciations per word, with and without counts, as well as on dictionaries extended with variants generated by a state-of-art data-driven method. This measure correlates well with our intuition about how the confusability is influenced by the change of the recognition lexicon. The baseline dictionary including the most-frequent pronunciations introduces a higher confusability than the baseline dictionary that includes the canoncical pronunciations, because the most frequent pronunciations are often reduced types that can be more easily confused. In addition, adding more variants to the recognition dictionary results in an augmentation of the entropy, since the number of homophones (or quasi-homophones) increases.

We would like to highlight the fact that the pronunciation entropy introduced in this work is a measure of the confusability at the sentence level. This is novel, and important because when adding many variants to an ASR system what is more important than the

homophone rate in the dictionary, is to measure this rate in the data. In fact the homophone rate in the original recognition dictionary (baseline) is 1.16, while in the baseline "longest" it is 1.10 and in the baseline "most frequent" it is 1.15. When adding up to 5 pronunciation variants to the baseline "longest" or the baseline "most frequent", the homophone rate becomes 1.24 in both cases. All these rates are close to one another, so it seems that what mostly influences confusability are some frequent homophone words or word sequences.

It is not straightforward however to find a correlation between the confusability and the ASR performance. What makes this procedure particularly complicated is the fact that confusable words are a non-negligible phenomenon of natural speech and ignoring them severely reduces the completeness of the dictionary, meaning that a consistent set of pronunciations is not necessarily connected with a pronunciation network of low perplexity. This lack of correlation turned our interest at this point towards other techniques that would allow us to add variants to a recognition dictionary without degrading the ASR performance. That is why we started experimenting with discriminative techniques for the construction of a dynamic lexicon using also phonemic information provided by the speech data. This is the work that will be presented in the following chapters.

Chapter 5

Discriminative training of a phoneme confusion model for a dynamic lexicon in ASR

In the previous chapter, we analyzed and measured the pronunciation confusability. The proposed measure, however, was not found to be clearly correlated with ASR performance. Until that point, the focus was on constructing and analyzing the performance of a pronunciation dictionary without using any specific information provided by speech data. That was the case in g2p and p2p conversion, where our training corpus was a given lexicon. That was also the case in our study on measuring the pronunciation confusability using the information provided by the linguistic part of the ASR decoding (pronunciation and language models). Because of the lack of improvement of the ASR performance using such approaches, we decided to turn towards approaches that will allow us to construct a dynamic lexicon trained on a particular audio data set, as is traditionally done, for example, for the acoustic model training. The goal is to add pronunciation variants with weights trained to directly improve the error rate of the ASR system. A discriminative framework is chosen to achieve this goal. The objective function to minimize is the phoneme error rate, directly related to the system's performance.

5.1 Introduction

As already analyzed in the Background Section 2.2, while all the other parts of an ASR system are trained to be adapted to particular data, this is not often the case for the components (i.e pronunciations) and the weights of the recognition dictionary. There have been some efforts, however, over the last years in this direction. In (Weintraub et al., 1996), the best performance is achieved by extracting pronunciation probabilities based on the frequency counts of each word and pronunciation. This can be applied only to words present in the training set and no further training of the weights is performed. Another method proposed in (Shu and Lee Hetherington, 2002), (Hazen et al., 2005) and (Badr et al., 2010) is the EM training of the weights of the lexicon. Nevertheless, this generative method often suffers from over-fitting to the training data. That is why in

the last years there is a turn towards discriminative methods. In (Vinyals et al., 2009), maximum entropy is used to determine the weight for each pronunciation. In a somehow different approach, discriminative training is also used by (Jyothi et al., 2012) to train an articulatory feature-based model of pronunciation. The drawback is that such methods are often computationally expensive and, thus, are tested on small data sets. Moreover, the latter work is once again limited only to words present in the training set.

In this work, we adapt a discriminative framework for training the weights of the pronunciation model and we conduct experiments on large data sets to evaluate the proposed method in a real-world task. First, the output of a phoneme recognizer is aligned with the reference and a set of phoneme confusion pairs is extracted. These confusion pairs form a confusion model that will be used to expand the phonemic search space of pronunciations during the ASR decoding. To train the weights of this confusion model, we compose it with the output lattices of the phoneme recognizer and a discriminative training is effected minimizing the phoneme edit distance between the output sentences of the phoneme recognizer and the true reference sentences. This results in a set of phonemic rules, that can be applied on the baseline pronunciations. In this way we hope to have pronunciations that better reflect the actual spoken utterances (see Section 5.6). Note that in this way we do not add a fixed number of pronunciations per word, as done with static g2p conversion.

As mentioned above, discriminative training is used to train the confusion model's weights, which aims to directly improve the system's performance while keeping the confusability low. In a discriminative model, the parameters of the model (i.e., pronunciation or language model probabilities) are adapted to minimize the recognition error rate. By contrast, the parameters of a maximum likelihood model are derived, as the name suggests, to maximize the likelihood of some data given the model; an increase in the likelihood of training data, however, does not always translate into reduced error rates.

Another way of seeing the application of our confusion model is as a corrector of the errors of the phoneme recognizer. The study of (Greenberg et al., 2000) has shown that phonetic and word errors are correlated (see Section 2.2), a fact that justifies our choice of an objective function in the phoneme level. Working at the phoneme level allows us to add variants to the baseform pronunciations of any word and not be limited to words that are present in the training data. Moreover, there are less parameters to train (limited number of phoneme pairs) compared to searching the confusions at the word or the sentence level and, thus, we are able to work with a continuous speech ASR system segmented in long sentences (approx. 80 words/sentence, see Section 5.5).

The assumption that improving the PER can result in suitable pronunciation variants for a word-based system can be reasonable under certain conditions. The acoustic model of a LVCSR system is trained with certain pronunciations that may not be correct, but once it is trained this way, correcting the pronunciations can do more harm than good. Consider an extreme case where pronunciations of all words were generated by a g2p converter making a consistent error such as switching phonemes /a/ and /e/. When the acoustic model is trained, it learns that every /a/ sounds like /e/ and the other way around, but since all the pronunciations are consistent with this error, it can still give perfect word recognition. Introducing a confusion model that would correct the pronunciations by switching /a/ and /e/ would actually break the system. Of course, we do not have such

extreme errors in our pronunciation dictionaries, but there certainly will be some errors in the pronunciation dictionary, and these errors will be still consistent with the acoustic model that was trained using it. However, this problem is alleviated if the reference phoneme sequence for confusion model training is obtained from the reference word sequence using forced alignment with the same pronunciation dictionary. This way, the confusion model accounts for differences between what phoneme sequence the phoneme recognizer tends to recognize and what is the "correct" phoneme sequence according to the original unexpanded pronunciation dictionary. Nevertheless, to really benefit from this confusion model, it might be profitable to retrain the acoustic model after applying the confusion model (or with the new pronunciation dictionary generated using the confusion model), which can be time-consuming though and is not done in this work.

For the moment, a simple unigram phonemic confusion model is used. This work is presented as a proof of concept that this method and its possible expansions (for ex. a direct generalization to the case of a contextual confusion model is possible) can be promising for the adaptation of the recognition dictionary to a particular data set. We present experiments using the averaged perceptron, originally proposed by (Freund and Schapire, 1999), and the CRF model, originally proposed by (Lafferty et al., 2001). The acoustic scores are used during the training of the pronunciation weights, enabling us to integrate some phonemic information provided by the acoustic model. This can improve the results as observed in (Weintraub et al., 1996) and (McGraw et al., 2013). All the implementations are done with Finite State Transducers (FSTs). For basic background information on the theory of FSTs the reader is directed to Section 2.3.

5.2 Problem set-up

We train a phoneme confusion model that acts as an error corrector of the output of the phoneme recognizer. In this work, as already mentioned, for the sake of simplicity a unigram model of phoneme pairs including substitutions and deletions is trained. The training algorithms used maximize a convex objective function and, thus, no specific initialization of our confusion model is necessary.

Formally define inputs and outputs: Let $\mathcal{X}$ denote an input space and, for a particular $x \in \mathcal{X}$, let $\mathcal{Y}(x)$ denote an output space. Given an $x \in \mathcal{X}$ and a $y \in \mathcal{Y}(x)$, $f(x, y)$ denote a feature vector representation of x and y and θ is a parameter vector containing one component for each feature.

We assume a training set consisting of n examples $\{\langle x^{(i)}, y^{(i)} \rangle\}_{i=1}^{n}$, where $x^{(i)}$ is a phoneme lattice and $y^{(i)}$ the reference corresponding to the true phoneme sequence. The phoneme lattice can be expanded with the use of the confusion model. Let $\mathcal{Y}(x^{(i)})$ be the set of phoneme sequences in the expanded phoneme lattice. Each path π in $\mathcal{Y}(x^{(i)})$ corresponds to an alignment (x, y) of a recognized phoneme sequence x and a possible correction y.

Decoding/Inference:The phoneme decoding problem requires solving

$$y^* = \arg\max_{y' \in \mathcal{Y}(x)} \theta^\top f(x, y'). \tag{5.1}$$

Decoding thus amounts to choosing the minimum-scoring path in the FST representing $\mathcal{Y}(x)$. The features $f(x, y)$ correspond to the phonemic confusions represented by our model. By changing the weights θ, we also change the path weights and, thereby changes the best path in the FST. Discriminative training aims at changing the weights θ in such a way that the best path gets closer to the reference phoneme sequence. This approach can be viewed as a discriminative "reranking" of the pairs $f(x, y)$ that occur in $\mathcal{Y}(x)$.

5.3 Training criteria

A critical step before decoding is the computation of the optimal weights θ. The weights are chosen to maximize a scoring function. For the sake of completeness, we review four criteria for training the weights θ. However, only two of them are implemented in the present work (the perceptron and the CRF model). The other two are mentioned as possible extensions to this work.

5.3.1 The CRF model

As a first training criterion, we can use the conditional log-linear model. Following the notations of (Gimpel and Smith, 2010), the conditional log-linear (CLL) model is defined as

$$p_\theta(y|x) = \frac{\exp\{\theta^\top f(x, y)\}}{\sum_{y' \in \mathcal{Y}(x)} \exp\{\theta^\top f(x, y')\}}. \tag{5.2}$$

A small modification is needed though. In addition to the weights θ of the confusion model, there also exist the scores $a_{x,y}$ from the acoustic model of the phoneme sequences x propagated to the sequence y. These scores are independent of θ and appear as an additive factor in Equation 5.2. Since they do not depend on θ they do not contribute to the derivatives as we will see below and, therefore do not interfere with the optimization. The CRF model becomes

$$p_\theta(y|x) = \frac{\exp\{\theta^\top f(x, y) + a_{x,y}\}}{\sum_{y' \in \mathcal{Y}(x)} \exp\{\theta^\top f(x, y') + a_{x,y'}\}} \tag{5.3}$$

The corresponding problem of training the weights θ by maximizing the conditional log-likelihood can be expressed as

$$\max_\theta \sum_{i=1}^{n} \big[\theta^\top f(x^{(i)}, y^{(i)}) + a_{x^{(i)},y^{(i)}} - log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^\top f(x^{(i)}, y) + a_{x^{(i)},y}\}\big] \tag{5.4}$$

Note that for the time being, no regularization term is used in the CRF model. Later we plan to experiment on using L_2 and L_1 regularizations. The L_2 penalty term $\frac{C}{2}||\theta||_2^2$ is the standard approach for parameter estimation in CRFs, where C is a regularization parameter. It smoothes the objective function to avoid over-fitting. The most significant alternative to L_2 regularization is to use the L_1 penalty term $C_1||\theta||_1$:such regularizers are able to yield sparse parameter vectors in which many component have been zeroed and,

thus, implicitly perform feature selection (Tibshirani, 1996). The combination of the two penalty terms is also possible (Zou and Hastie, 2005). It could also be interesting to use an entropy-based regularization term (Grandvalet and Bengio, 2006) (see the Conclusion section 5.8).

The CRF training criterion is equivalent to MMI training traditionally used in speech recognition to discriminatively train the acoustic model's weights (Povey, 2003). It could be argued that this is a complicated model whose power is not fully used in our case of a unigram context-independent confusion model. However, the aim is to develop a framework that can be later generalized to more complicated features without any changes.

5.3.2 Soft-margin CRF

Instead of using the CRF model, the softmax-margin objective is an alternative based on the simple intuition that high cost outputs for $x^{(i)}$ should be penalized more heavily. The advantage of this objective compared with the CRF model is the ability to incorporate task-specific cost functions. The soft-margin CRF objective to maximize is

$$f(\theta) = \sum_{i=1}^{n} \big[\theta^\top f(x^{(i)}, y^{(i)}) + a_{x^{(i)},y^{(i)}} - log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^\top f(x^{(i)}, y) + a_{x^{(i)},y} + cost(y^{(i)}, y)\}\big], \tag{5.5}$$

where $cost(y^{(i)}, y)$ a task-specific cost function that measures the extent to which a structure y differs from the true structure $y^{(i)}$.

Equation 5.5 is a generalization of Povey's Boosted MMI (Povey et al., 2008). It can be seen as replacing the maximum of max-margin (see Section 5.3.3.2) by a soft margin that is proportional to the number of errors in the hypothesized sentence. It can be shown that it bounds the minimum risk optimization criterion (Gimpel and Smith, 2010). This variation also yields a convex optimization problem which can be efficiently solved using standard tools. It can be implemented atop standard CRF training and requires the same amount of computation as the CRF model.

5.3.3 Large-margin methods

Instead of a probabilistic view that transforms $\theta^\top f(x, y)$ into a probability, we take a geometric view in which $f(x, y)$ is an embedding of $\langle x, y \rangle$ into R^d. During training, we consider the point $f(x^{(i)}, y^{(i)})$ and all competing points $f(x^{(i)}, y)$ for $y \in \mathcal{Y}(x^{(i)}) \backslash \{y^{(i)}\}$. The goal is to choose a direction (encoded in the weight vector θ) along which the point $f(x^{(i)}, y^{(i)})$ dominates its competitors, i.e., $\theta^\top f(x^{(i)}, y^{(i)})$ has a high value. Furthermore, the alternative points $f(x^{(i)}, y)$ should all receive scores $\theta^\top f(x^{(i)}, y)$ that are inversely proportional to the amount of error incurred in labeling $x^{(i)}$ with y when the true answer is $y^{(i)}$. This is naturally encoded in a cost function $cost(y^{(i)}, y)$ as already mentioned.

5.3.3.1 Perceptron

The perceptron can be seen as an approximation of the online version of the CRF training criterion if we approximate the posterior probability of the most likely hypothesis

to one and all the other hypotheses to zero. The perceptron algorithm iteratively updates weights by considering each training example in turn. On each round, it uses the current model to make a prediction. If the prediction is correct, there is no change to the weights. If the prediction is incorrect, the weights are updated proportionally to the difference between the correct feature vector $f(x^{(i)}, y^{(i)})$ and the predicted feature vector $f(x^{(i)}, y^*)$. Following the perceptron algorithm as presented in (Smith, 2011), the weight update for each training example is:

$$\theta \leftarrow \theta + a\big(f(x^{(i)}, y^{(i)}) - f(x^{(i)}, y^*)\big), \tag{5.6}$$

where a is the learning rate. This is a variant of the perceptron algorithm also applied to language modeling in (Roark et al., 2004).

The actual loss function of the perceptron that we search to minimize is the following approximation to the zero-one loss:

$$\frac{1}{n}\sum_{i=1}^{n} \theta^\top \big(f(x^{(i)}, y^{(i)}) - f(x^{(i)}, y^*)\big). \tag{5.7}$$

We introduce here the notion of margin that we will need later. In separable problems, perceptron can also aim at finding the largest separating margin between the classes:

$$\theta^\top \big(f(x^{(i)}, y^{(i)}) - f(x^{(i)}, y)\big) > \delta \quad \forall i, y, \tag{5.8}$$

where $\delta > 0$ the margin with which the data are separable, if there exists a weight vector θ that satisfies relation 5.8. If the data are separable, there is theoretical evidence of the convergence of the algorithm (Collins, 2002).

Also following (Collins, 2002), we use the averaged parameters from the training algorithm in decoding the held-out and test examples. Say $\theta_t^{(i)}$ is the parameter vector after the ith example is processed on the t pass through the training data. Then the averaged parameters are defined as $\theta_{AVG} = \sum_{i,t} \theta_t^{(i)}/nT$, where n is the number of examples in our training set and T the number of passes over the training set. The *averaged perceptron* was originally proposed by (Freund and Schapire, 1999) and was shown to give substantial improvements over the non averaged version in accuracy for tagging tasks (Collins, 2002).

5.3.3.2 Max-margin

When adding a cost function to the margin, Equation (5.8) that expresses the margin constraints becomes

$$\theta^\top \big(f(x^{(i)}, y^{(i)}) - f(x^{(i)}, y)\big) \geq cost(y^{(i)}, y) \quad \forall i, y. \tag{5.9}$$

The learning problem becomes the minimization of the objective respecting the constraints of Equation (5.9). Let the objective be the L_2 penalty:

$$\min_\theta \frac{C}{2}||\theta||_2^2 \tag{5.10}$$

If we relax the constraints,

$$\min_{\theta,\xi} \frac{C}{2}||\theta||_2^2 + \sum_{i=1}^{n} \xi_i \tag{5.11}$$

$$s.t.\ \xi_i \geq -\theta^\top f(x^{(i)}, y^{(i)}) + \max_{y \in \mathcal{Y}(x^{(i)})} \big(\theta^\top f(x^{(i)}, y) + cost(y^{(i)}, y)\big). \tag{5.12}$$

Thus, the objective function can be written as

$$\min_{\theta} \sum_{i=1}^{n} \big[\frac{C}{2}||\theta||_2^2 - \theta^\top f(x^{(i)}, y^{(i)}) + \max_{y \in \mathcal{Y}(x^{(i)})} \big(\theta^\top f(x^{(i)}, y) + cost(y^{(i)}, y)\big)\big]. \tag{5.13}$$

To this term we should not forget to add the score from the acoustic model as we did before for the CRF model:

$$\min_{\theta} \sum_{i=1}^{n} \big[\frac{C}{2}||\theta||_2^2 - \theta^\top f(x^{(i)}, y^{(i)}) - a_{x^{(i)},y^{(i)}} + \max_{y \in \mathcal{Y}(x^{(i)})} \big(\theta^\top f(x^{(i)}, y) + a_{x^{(i)},y} + cost(y^{(i)}, y)\big)\big]. \tag{5.14}$$

5.3.4 Optimization algorithm

For CRF and soft-margin CRF a gradient descent with learning rate a can be used as an optimization algorithm. The derivatives that need to be computed for the CRF are:

$$\frac{\partial CRF(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \big[f_j(x^{(i)}, y^{(i)}) - \sum_{y \in \mathcal{Y}(x^{(i)})} f_j(x^{(i)}, y) p_\theta(y|x^{(i)})\big] \tag{5.15}$$

$$= \sum_{i=1}^{n} \big[f_j(x^{(i)}, y^{(i)}) - \mathbb{E}_{p_\theta(y|x^{(i)})}[f_j(x^{(i)}, y)]\big] \tag{5.16}$$

The feature expectation $\mathbb{E}_{p_\theta(y|x^{(i)})}[f_j(x^{(i)}, y)]$ is the averaged value of the feature f_j across all $y \in \mathcal{Y}(x^{(i)})$, with each y weighted by its conditional probability given $x^{(i)}$.

Using the log-linear form of the model (Equation (5.3)), the expectation rewrites as:

$$\mathbb{E}_{p_\theta(y|x^{(i)})}[f_j(x^{(i)}, y)] = \frac{\sum_{y \in \mathcal{Y}(x^{(i)})} f_j(x^{(i)}, y) \exp\{\theta^\top f(x^{(i)}, y) + a_{x^{(i)},y}\}}{\sum_{y' \in \mathcal{Y}(x^{(i)})} \exp\{\theta^\top f(x^{(i)}, y') + a_{x^{(i)},y'}\}}. \tag{5.17}$$

Let $Z_{x^{(i)}} = \sum_{y' \in \mathcal{Y}(x^{(i)})} \exp\{\theta^\top f(x^{(i)}, y') + a_{x^{(i)},y'}\}$ be the normalization term, independent of y. The expectation is computed using the standard forward-backward algorithm.

The additional computation involved in the soft-margin CRF compared to the CRF is very small: when applying the forward-backward algorithm to compute the features expectation, we take into account also the cost of each hypothesis. More details will be given in the FST implementation section (see Section 5.4).

For the perceptron, its standard update formula is used as already mentioned. For the max-margin, stochastic subgradient descent with a step a is used because the objective is not differentiable everywhere because of the $\max$ operation. Let $MM(\theta)$ be the max-margin objective presented in Equation (5.14). A subgradient of this objective is:

$$\frac{\partial MM(\theta)}{\partial \theta_j} = C\theta_j - f_j(x^{(i)}, y^{(i)}) + f_j \left(x^{(i)}, \arg\max_{y \in \mathcal{Y}(x^{(i)})} \left(\theta^\top f(x^{(i)}, y) + a_{x^{(i)},y} + cost(y^{(i)}, y) \right) \right) \tag{5.18}$$

An additional comment regarding CRF training is in order: until now we presented a simple supervised learning setup where learning is done with gradient descent. However, in this work online training is chosen, which is computationally less expensive than batch training. In online classification, the model is updated after each example. In this case, the term iteration of the training algorithm corresponds to processing one training example and making an update. We refer to an iteration over the whole training set with the term epoch. The gradient descent is thus replaced by stochastic gradient descent, a simplified algorithm. Meaning that each iteration estimates this gradient on the basis of a single randomly selected example (Bottou, 2010). In the perceptron case, the stochastic gradient descent matches the original algorithm.

The learning rate is the rate of convergence of the training algorithms. If the step size is too small, the convergence is slower. If the step size is too large, the weights may oscillate and not reach a stable point. In online training, it has been found that it is better not to use a fixed learning rate a. Instead, learning rates are generally decreased according a schedule of the form $a = a_0/(1 + a_0 * t)$, where $t = 1, 2, ...n$ is the iteration of the learning algorithm (the example we are processing). This schedule was originally proposed by (Robbins and Monro, 1951). It is a gradually decaying learning rate, but smoother than $1/t$. The initial rate a_0 was heuristically set to $a_0 = 0.1$.

5.4 An FST-based implementation

In the following section, we use some abbreviations to refer to the basic FST algorithms (see Section 2.3.5): ShortestPath(), ShortestDistance(), RmEpsilon() (remove all ϵ-transitions), Determinize(), Minimize(), Push() (push weights), π_1() (project to the input omitting output labels) and π_2() (project to the output omitting input labels).

5.4.1 Preprocessing

We take the phoneme lattices generated from the phoneme recognizer described in Section 5.5. To avoid the problem of duplicated hypothesis in the input lattices since no time information is kept, pauses, fillers and silences are removed both from the input lattices and the reference lattices in a preprocessing step. We then apply the RmEpsilon(), Determinize() and Minimize() algorithms on the lattices (see Section 2.3.5). Thus, in each lattice each input sentence is represented by exactly one path.

The applied algorithms are optimization algorithms, which optimize the time and/or space requirements of the weighted transducers on which they are applied. The ϵ-transitions

cause some delay in the use of the resulting transducers since the search for an alphabet symbol to match in composition or other similar operations requires reading some sequences of ϵs first. Concerning determinization, the basic advantage of a deterministic automaton is the absence of redundancy: it contains at most one path matching any given input sentence, thereby reducing the time and space needed to process an input sequence. Finally, a deterministic weighted automaton which is minimized has the smaller number of states among the deterministic automata that realize the same computation (i.e., are equivalent).

5.4.2 Defining the input and output FSTs

Depending on the task, we work in the log or in the tropical semiring. In these semirings all arc weights represent negative log weights that can be interpreted as costs.

Let Ph be the input lattice after the preprocessing and R the reference phoneme string represented as an FSA. Ph is a lattice with arc weights that represent the acoustic scores of the phoneme recognizer. It is not probabilistic. The paths (phoneme sequences) in this lattice are the input x to the discriminative training.

In this work, a unigram model of phoneme confusions (substitutions and deletions) is used, represented by an FST $C(\theta)$, where the vector θ has one component for each possible confusion. $C(\theta)$ is an one-state FST whose structure is designed as follows. We first run a forced alignment of the training data with the reference, yielding phonemic references; we then align the one-best outputs of the phoneme recognizer with the corresponding reference sequences and count the number of phoneme deletions and substitutions. Confusion pairs that appear less than 20 times are not kept to avoid learning hazardous mistakes [1]. The resulting FST contains 1021 phoneme pairs, for which weights are to be trained. Each confusion is represented by one arc in the FST, whose input (resp. output) symbol corresponds to the recognized (resp. reference) symbol. Thus, each arc expresses a phoneme substitution, deletion or identity (if there is not a misrecognition of the reference phoneme).

We compose $\mathrm{Ph} \circ C(\theta)$ to expand the input lattice. Let's call $\mathrm{Ph}_1(\theta)$ the resulting FST. The sequence of input symbols of a path in the new FST is the x and the sequence of its output symbols is the y of the feature vector representation $f(x, y)$. Let $\mathcal{Y}(x)$ be the set of all paths projected to the output in $\pi_2(\mathrm{Ph}_1(\theta))$. Let $y^{(i)}$ be the reference corresponding to the true phoneme sequence or the closest approximation of y to the true phoneme sequence. If the ShortestPath() algorithm on the tropical semiring is applied on $\mathrm{Ph}_1(\theta)$, let y^* be the output phoneme sequence corresponding to the best path of the FST.

Note that the arc weights of $\mathrm{Ph}_1(\theta)$ result from the composition of the acoustic scores of Ph and the confusion model's scores of $\mathrm{C}(\theta)$. Since we work on the log semiring, the path weight is computed by summing the weights of the arcs of the path. In these preliminary experiments, we are using a unigram confusion model, meaning that each arc weight is independent of the weights of the other arcs.

Decoding becomes the problem of choosing the minimum-scoring path on the tropical semiring in the FST representing $\mathcal{Y}(x)$. The weights θ should be trained such as the

1. This lowest limit of appearance of the confusion pairs was heuristically set to 20.

best path y^* resulting from decoding is the one whose distance from the reference is minimized.

5.4.3 Computing the edit distance with FSTs

To compute the loss function, we need to compute the minimum edit distance between the reference $y^{(i)}$ and the surface phonemic sequences of the lattice $x^{(i)}$. To do so, an efficient way is provided with the use of FSTs. The edit distance is defined as:

$$editdistance = \#ofsubstitutions + \#ofdeletions + \#ofinsertions$$

In our case, we compute it at the phoneme level, and refer to it as a phoneme edit distance. We actually compute the Levenshtein distance as all costs are set to unity except for matches which have zero cost. Following the definition of (Heigold, 2010),

$$d(x^{(i)}, y^{(i)}) = min_{x \in x^{(i)}, y \in y^{(i)}} d(x, y), \tag{5.19}$$

where x (resp. y) is a phonemic sequence of the set of phonemic sequences of the lattice $x^{(i)}$ (resp. $y^{(i)}$). The reference $y^{(i)}$ is represented as a lattice if multiple references are available. In this work, it is a single phonemic sequence. where x (resp. y) is a phonemic sequence of the set of phonemic sequences of the lattice $x^{(i)}$ (resp. $y^{(i)}$). The reference $y^{(i)}$ is represented as a lattice if multiple references are available. In this work, it is a single phonemic sequence.

The edit distance transducer LD is a WFST that defines the phoneme alignments with their costs, i.e. each arc represents a local edit operation with the respective cost as the arc weight. Deletions and insertions are encoded with the empty symbol ϵ (used in the hypothesis' side for deletions and in the reference's side for insertions).

If $x^{(i)}$ and $y^{(i)}$ are represented as unweighted FSTs, then the edit distance can be computed on the tropical semiring as:

$$d(x^{(i)}, y^{(i)}) = ShortestPath(x^{(i)} \circ LD \circ y^{(i)}) \tag{5.20}$$

5.4.4 Discriminative training algorithms

In Section 5.4.2, we explained how the best path y^* in $\mathrm{Ph}_1(\theta)$ is found and that this is the path we would like to approach to the reference. The next step is to find the path in $\mathrm{Ph}_1(\theta)$ which has the minimum distance to the reference. To do so, we introduce an FST which represents all possible editions between $\mathrm{Ph}_1(\theta)$ and the reference R. This is computed by the operation $E = \mathrm{Ph} \circ \mathrm{C} \circ LD \circ R$ on the tropical semiring, where $\mathrm{Ph} \circ \mathrm{C}$ has no weights. The operation ShortestPath(E) gives us the path $y^{(i)}$ in $\mathrm{Ph}_1(\theta)$ with the minimum distance from the reference. Updating the weights θ should get us closer to the path $y^{(i)}$. Another way of viewing it is that y^* is the best hypothesis given θ and $y^{(i)}$ is the optimal hypothesis.

Next, the training algorithms for the different optimization criteria are presented and a pseudocode is provided for each case.

5.4.4.1 Perceptron

We start with a variant of the perceptron algorithm:

for t=1...T, i=1...n **do**
  Compute $y^{(i)}$, then $\langle x^{(i)}, y^{(i)} \rangle$
  Compute y^*, then $\langle x^{(i)}, y^* \rangle$
  if $\langle x^{(i)}, y^{(i)} \rangle \neq \langle x^{(i)}, y^* \rangle$ **then**
    $update = \langle x^{(i)}, y^{(i)} \rangle - \langle x^{(i)}, y^* \rangle$
    $\theta \leftarrow \theta + a * update$
  end if
end for

where $\langle x, y \rangle$ is the feature representation in the corresponding paths and T the number of epochs.

5.4.4.2 Max-margin

For the max-margin training algorithm, the computation of

$$f_j \left(x^{(i)}, \arg\max_{y \in \mathcal{Y}(x^{(i)})} \big(\theta^\top f(x^{(i)}, y) + a_{x^{(i)},y} + cost(y^{(i)}, y) \big) \right)$$

is needed. We have to add the cost corresponding to each arc of $\mathrm{Ph}_1(\theta)$ and then find the best path. This cost is the distance from the reference. To have an FST equivalent to the $\mathrm{Ph}_1(\theta)$ but with the cost added to each arc, the following operation is performed: Minimize(Determinize($\pi_1(\mathrm{Ph}'_1(\theta) \circ LD \circ R)$)). Then, the ShortestPath() algorithm is applied. Note that to keep both the input and output symbols of $\mathrm{Ph}_1(\theta)$ after the above composition, a mapper is applied on $\mathrm{Ph}_1(\theta)$. The new FST $\mathrm{Ph}'_1(\theta)$ uses both input and output symbols of $\mathrm{Ph}_1(\theta)$ as a combined input, while its output symbol is the output symbol of $\mathrm{Ph}_1(\theta)$ unchanged. The training algorithm pseudocode is given below:

for t=1...T, i=1...n **do**
  Compute $\langle x^{(i)}, y^{(i)} \rangle$
  Compute $\langle x^{(i)}, y^*_{cost-augmented} \rangle$
  $update = \langle x^{(i)}, y^{(i)} \rangle - \langle x^{(i)}, y^*_{cost-augmented} \rangle$
  $\theta \leftarrow \theta + a * update$
end for

5.4.4.3 CRF

For the CRF training, we use a stochastic gradient descent with learning rate a. As explained in Section 5.3.4, we use online training instead of computing the objective and the derivatives on the whole corpus. The derivatives are computed in Equation (5.15). We need to compute the expectation of each feature:

$$\mathbb{E}_{p_\theta(y|x^{(i)})}[f_j(x^{(i)}, y)].$$

To do this, the forward-backward algorithm is applied on the FST $\mathrm{Ph}_1(\theta)$. The forward values (from the initial state to each state q) $alpha[q]$ and the backward values (from each state to the final state q) $beta[q]$ are computed with the ShortestDistance() operation. Both the forward and the backward values are stored as vectors. Then, the expectation is computed as follows:

for StateIterator q **do**
  for ArcIterator k from each state q **do**
    $arc.weight_k = \exp\{\theta^\top f(x_k^{(i)}, y_k) + a_{x_k^{(i)}, y_k}\}$
    Compute $p_\theta(y_k|x_k^{(i)}) = \frac{alpha[q]*beta[arc.nextstate]*arc.weight_k}{Z_{x^{(i)}}}$
    $\mathbb{E}_{p_\theta(y_k|x_k^{(i)})} = p_\theta(y_k|x_k^{(i)}) * < x_k^{(i)}, y_k >$
    $\mathbb{E}_{p_\theta(y|x^{(i)})} += \mathbb{E}_{p_\theta(y_k|x_k^{(i)})}$
  end for
end for

The normalization term $Z_{x^{(i)}}$ is computed with a ShortestDistance() (backward) operation to the initial state. Instead, we can normalize the probabilities of $\mathrm{Ph}_1(\theta)$ with the Push(, REWEIGHT_TO_INITIAL) operation before applying the ShortestDistance() operations. In this case, there is no need for the term $Z_{x^{(i)}}$ in the just above described algorithm for the expectations' computation. All ShortestDistance() operations are performed on the log semiring.

Once the vector of expectations for all features $\mathbb{E}_{p_\theta(y|x^{(i)})}[f(x^{(i)}, y)]$ is computed, the CRF training algorithm can be applied:

for t=1...T, i=1...n **do**
  Compute $\langle x^{(i)}, y^{(i)} \rangle$
  Compute $\mathbb{E}_{p_\theta(y|x^{(i)})}[f(x^{(i)}, y)]$
  $update = \langle x^{(i)}, y^{(i)} \rangle - \mathbb{E}_{p_\theta(y|x^{(i)})}[f(x^{(i)}, y)]$
  $\theta \leftarrow \theta + a * update$
end for

5.4.4.4 Soft-margin CRF

The only difference between the CRF and the Soft-Margin CRF training is that the forward-backward algorithm is applied this time on the FST resulting from the operations Minimize (Determinize($\pi_1(\mathrm{Ph}_1'(\theta) \circ LD \circ R)$)). Other than that, the training algorithm is the same as in the previous section.

5.5 Experimental setup

The phoneme recognizer used in this chapter is the same as the one described in Section 4.3. We use the output of a phoneme recognizer to get the surface phonemic transcriptions and then align them to the canonical transcription (reference). In this way we get a set of phoneme confusion pairs. A phonemic 3-gram language model is used in the construction of the phoneme recognizer to impose some constraints in the generated

phonemic sequences. At the other extreme, a phone-loop recognition with no grammar would produce totally unbiased phonemic strings, but these strings may not form valid candidates according to the lexicon.

For discriminative training we have 40h of broadcast news (BN) data used by the Quaero project (www.quaero.org), which include 5k phoneme lattices. Lattices with very high error rate were removed and the remaining 4k lattices were used for training. The removed lattices presented a very high error rate because of lack of reference for the particular time segments or because of other unpredictable factors (i.e. extreme presence of noise,...). The PER on the training data is 35%. Note that we are working with real-world continuous speech, segmented in particularly long sentences (on average 80 words/sentence). The phoneme lattices used for training have on average 1,000 phonemes/sentence, with the actual length of each lattice varying from 2 to 10,000 phonemes per sentence.

The Quaero dev2010 (4h) were evenly subdivided into test and dev sets. These are the data used as test data in the previous chapters of this thesis (see Sections 3.5 and 4.3). The test and the dev sets each contain 350 lattices. These data are more difficult to handle than standard BN data, because there are mixed with broadcast conversational (BC) data.

An FST decoder is needed for some of the experiments presented in Section 5.7. It is the same as the one used in Chapter 4. Is is a simple one-pass decoder that does not take into account word boundaries or time information.

5.6 Phonemic analysis

At this point, the most common confusions learned after the training of our confusion model warrant some further analysis. At the end of the first epoch of the CRF model training (4k iterations of the online training algorithm), the confusions learned seem to be compatible with phenomena encountered in continuous speech. Common substitutions include the syllabic consonants (see Appendix A) being substituted by the nearest consonant (ex. */N/* by */n/* or */L/* by */l/*). There are also observed reductions to schwas (ex. */N/* to */x/*, */U/* to */x/* or */R/* to */X/*). Moreover, some vowels that represent similar sounds are getting confused, for ex. */U/* by */c/* and */W/* by /@/. This is the case for consonants, too, for ex. */N/* getting substituted by */d/*, */S/* by */T/* and */L/* by */r/*. The most frequent deletions are the deletions of */N/*, */R/* and */U/*, which can be easily deleted when unstressed. Note that some of these changes (for ex. substitution of syllabics by their corresponding simple consonant) are actually taken into account when restricting the used phoneme set. The restriction of the phoneme set to one with less phonemes is based on the merging of some phonemes that are responsible for some very current confusions. This method, originally proposed by (Lee and Hon, 1989) is often used to evaluate a phoneme recognizer's performance. This phoneme analysis provides a further indication that our model learns indeed some common confusions of continuous speech.

5.7 Evaluation

5.7.1 Computation of the objective

A first test that was realized to control that the discriminative training works properly was to compute the objective function that we try to optimize on the training data. As already mentioned, online training is chosen for the CRF case and for the perceptron, for which it is the default approach. Only one epoch over the training data is performed, since we would like to see the performance of the system keeping the time of computation low. This is why we actually chose to use online training which has been shown to be asymptotically efficient after a single pass on the training set (Bottou, 2010) [2]. The objective is computed after each 50 iterations (examples) on a randomly chosen sub-set of the training data set.

In the case of the perceptron, the loss function is an approximation of the zero-one loss given in Equation 5.7. This loss function, in the ideal case, should be zero if no difference between the best hypothesis and the reference was observed. In our case, as can be seen in Figure 5.1, the loss function converges to a minimum after around 1200 iterations of the training algorithm.

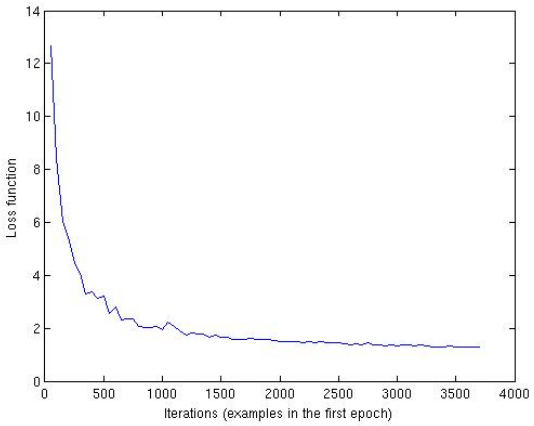

Figure 5.1: *Perceptron loss on training data*

In the case of the CRF model, we want to train the weights θ while maximizing the conditional log-likelihood (Equation 5.4). To see some improvement in the upper objective, some normalization of the initial acoustic weights a was necessary before combining them with the weights θ in order to have the weights in the same scale of values. After this normalization, the objective is indeed maximized as expected, as can be seen in Figure 5.2. Note that, for both perceptron and CRF, a convergence towards a stable point is shown in the respective figures within the first epoch on the training data.

2. We tried to train on several epochs but the PER on the dev set did not improve further, following the position on the convergence within the first epoch.

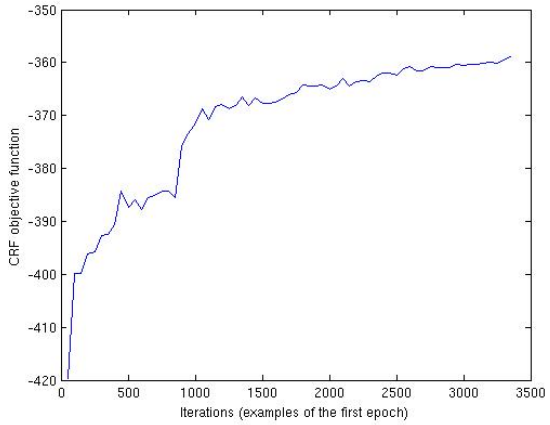

Figure 5.2: *CRF objective on training data*

Table 5.1: *Phoneme Accuracy in the dev set*

System	Correct(%)	Phoneme Acc(%)	Del(%)	Sub(%)	Ins(%)
Baseline	59.0	**54.9**	**19.1**	21.9	4.2
Perceptron	57.9	53.6	18.9	23.2	4.3
Averaged Perceptron	59.3	**55.0**	18.7	22.0	4.3
CRF	58.4	51.6	**15.9**	25.7	6.7

5.7.2 Phoneme Accuracy

Since the loss function we try to minimize is based on the phoneme error rate, it is interesting at this point to see if using a confusion model can improve the Phoneme Accuracy in the development (dev) and test sets. Slight improvements are observed over the baseline in the dev set for the Phoneme Accuracy (see Table 5.1). Note that the proposed model is a very simple one (a unigram model of substitutions and deletions) which can surely not capture exactly the phoneme context dependencies presented in pronunciation modeling. However, some partial improvements can be observed. For example, looking at the column "Deletions" in Table 5.1, the system with the CRF-trained confusion model reduces the deletion rate from 19.1% to 15.9%. The best performance is achieved by the averaged perceptron which slightly improves the phoneme accuracy.

Similar results are observed in the test set, which is a positive indication towards the generalization power of the chosen algorithms (see Table 5.2).

Table 5.2: *Phoneme Accuracy in the test set*

System	Correct(%)	Phoneme Acc(%)	Del(%)	Sub(%)	Ins(%)
Baseline	59.5	55.7	19.6	20.9	3.8
Perceptron	58.3	54.5	19.4	22.3	3.8
Averaged Perceptron	59.8	56.0	19.3	20.9	3.9
CRF	58.7	52.8	16.5	24.8	6.0

It is also interesting to note that adding the confusion model without any training of its weights severely degrades the system's performance. It results in a Phoneme Accuracy near zero and is not presented in any more details in the results tables. This is because of the expansion of the phonemic search space of pronunciations without any constraints, which adds a detrimental amount of confusability. There is an increase of 121% in the average number of paths in the phoneme lattices in the dev set after the composition with the confusion model and after some pruning is performed. An increase of 126% is observed for the test set. However, after the training of the weights the baseline performance is re-established and a slight improvement is observed. This result can be seen as a first indication that the weights of our model are trained in the right direction and it can be promising if more context is added in the model in the future.

Note also that the acoustic models we use are already context-dependent, plus a 3-gram phonemic LM is used in the phoneme recognizer. That means that a large part of the phonemic variation is already captured by the acoustic model and the phonemic LM. It would be maybe easier to see some improvements if a simple phoneme-loop phoneme recognizer was used to generate the phonemic lattice.

5.7.3 Decoding process

The next step is to introduce the confusion model into the decoding process of a word recognizer. We expand the phonemic search space of pronunciations and integrate with the pronunciation and the language models. Thus, instead of using a static recognition lexicon, a dynamically speech-adapted lexicon is produced.

To do so, an FST-based decoder is needed. The LIMSI state-of-the-art decoder is not based on the OpenFST implementation used to produce our confusion model. A first idea was to take the phoneme lattices produced by the phoneme recognizer and compose them with the confusion, pronunciation and language models. This would correspond to the decoding of the linguistic part of the system keeping the acoustic part stable (see Chapter 4). However, the PER of around 45% of the phoneme recognizer in the test set (see Table 5.2) is not a good starting point for this partial decoder. In fact, the resulted word error rate (WER) is very low with or without using the confusion model in the decoding procedure. Even the oracle WER is too low (around 50%) to allow a meaningful evaluation of our confusion model.

To circumvent this problem, we decided to add the confusion model in a post-processing step to the LIMSI word decoder. First, we take the 1-best word output of the LIMSI word recognizer and we express it as an FST W. Then we compose with the inverted FST of the pronunciation model Pr^{-1} and the result is a phoneme lattice A:

$$A = W \circ Pr^{-1}. \tag{5.21}$$

The pronunciation model FST Pr is the result of the composition of the FSTs $P \circ L$ (see Section 4.2.1) and assigns a sequence of phonemes to a word. At this point, the Phoneme Accuracy of this phoneme lattice is computed and compared with the performance of the phoneme recognizer. The Phoneme Accuracy of the phoneme lattice generated by the above mentioned composition is 70% (see the line "Lattice A" in the Table 5.3). The

Phoneme Accuracy of the phoneme recognizer in the same test set is 55% (as can be seen in the line "Phoneme recognition" in the Table 5.3). These results support our hypothesis that using the phoneme recognizer to generate word sentences fails because of the low quality of the phoneme recognizer.

This phoneme lattice is then expanded with the confusion model $C(\theta)$ and a new phoneme lattice B is generated:

$$B = W \circ Pr^{-1} \circ C(\theta). \tag{5.22}$$

The best Phoneme Accuracy of the expanded lattice B is found to be 77.8% when the confusion model is trained using the averaged perceptron (see line "Lattice B" with Training "CRF" in the Table 5.3), which corresponds to a large improvement over the baseline. These reslts were found to be significantly better ($p < 0.001$). Note that as baseline we consider the phoneme lattice A resulting for the 1-best output of the word recognizer with a simple composition with the reverse pronunciation lexicon (line "Lattice A" in the Table 5.3). Note also that there is a large improvement over the baseline for all three training criteria applied on the confusion model.

Observing more in detail the results of Table 5.3, it can be seen that the main improvement produced by the confusion model consists in an important decrease in the insertions, meaning that our confusion model learns to do more deletions. A discussion could be opened at this point on the utility of a unigram confusion model since some variation is already captured by the context-dependent phones (triphones) used in the acoustic modeling. There are, however, some cases were the triphones fail to model pronunciation variation, as shown by (Jurafsky et al., 2001). More in particular, (Jurafsky et al., 2001) showed that triphones cannot capture variation related to syllable deletions. This is the gap that our confusion model covers by an augmentation of the number of deletions that it provokes. Such a behaviour of our model is not observed, however, in Tables 5.1 and 5.2. This can be because of the lower baseline (lower phoneme accuracy) of the phoneme recognizer, meaning that more confusions are present that may be difficult to get consistently corrected even by our suitably trained confusion model.

Table 5.3: *Phoneme Accuracy of the phoneme output of the word recognizer (test set)*

	Correct(%)	Phoneme Acc(%)	Del(%)	Sub(%)	Ins(%)	Training
Phoneme recognition	59.5	55.7	19.6	20.9	3.8	
Lattice A	89.6	**69.9**	3.9	6.6	19.7	
Lattice B	81.3	**77.4**	6.5	12.2	3.9	CRF
	78.7	76.2	7.6	13.7	2.5	Perceptron
	81.0	**77.8**	7.3	11.7	3.2	Av. Perceptron

Next, we recompose with the pronunciation model Pr and the language model G (the same as the one used in Chapter 4) to produce a new word sequence W_1. To sum up, the series of compositions to get to W_1 is:

$$W_1 = W \circ Pr^{-1} \circ C(\theta) \circ Pr \circ G \tag{5.23}$$

This series of inverse compositions and recompositions is based on the idea presented in (Fosler-Lussier et al., 2005) implemented to find the confusable words and predict ASR errors. Ideally the new word sequence W_1 would have a lower word error rate compared to the word sequence W. However, the following problem occurs: comparing W and W_1 is not a fair comparison because they are not the outputs of the same decoder. Our FST decoder is surely a more simple one compared to the LIMSI decoder. It is an one-pass decoder, while the LIMSI decoder is a multiple-pass decoder using an n-gram language model of a higher order in each pass. Another weakness of our decoder is that no time information is kept and often there is not a perfect alignment of the decoder's output with the reference. If there are whole lattices missing from the reference we can of course ignore them during scoring, but this is not as simple when just parts of lattices are missing and no time information is kept in the decoder's output. Furthermore, in our simple FST-based decoder, no post-processing is applied on the data before the scoring (i.e. lowercase, remove symbols such as "-", correct the orthography of some words when there is a mismatch between the reference and the lexicon transcription,...). These are some standard normalizations performed on the data before the scoring of the output of the LIMSI decoder.

Moreover, since the inverted mappings are one-to-many (for ex. the lexicon Pr includes more than one pronunciations for certain words) and the word boundary information is lost after the compositions, the set W_1 will typically have more elements that W. There can be a lot of homophones in the new phoneme space after the composition with the inverted pronunciation model. And these homophones are not limited in within-word boundaries but can correspond to groups of words or even to whole sentences. Last but not least, the acoustic scores are lost during the inverse composition because there is not a way provided to keep the phoneme acoustic scores corresponding to the 1-best word output of the LIMSI decoder.

For the above reasons and for others concerning the detailed construction of the decoder, the baseline Word Accuracy of the FST decoder (before introducing the confusion model - see line "Lattice W_b" in Table 5.4) is lower than the one of the LIMSI decoder (around 70%). The lattice W_b is the result of the post-processing compositions:

$$W_b = W \circ Pr^{-1} \circ Pr \circ G. \tag{5.24}$$

As can be seen in Table 5.4, using the confusion model trained using the CRF model ("Lattice W_1" with Training "CRF") results in a slight improvement over the baseline ("Lattice W_b"), which is however statistically significant ($p < 0.02$). The perceptron and the averaged perceptron training criteria do not result in an improvement over the baseline. Furthermore, the large improvement observed in a phoneme level (Phoneme Accuracy improved from 70% to 77% for the CRF training, see Table 5.3) does not carry over to the level of words. This can be again because of the characteristics of the FST-decoder mentioned in the above paragraphs (the acoustic model's information is lost, no word-boundaries,...). It is not straightforward though how to integrate the FST-based confusion model to a non-FST decoder. It should also be noted that the training of the confusion model is not done in the same conditions; that is, the acoustic scores are used during the training. Doing the training on the phonemic transcription of the word output

Table 5.4: *Word Accuracy on the test set*

	Correct(%)	Word Acc(%)	Del(%)	Sub(%)	Ins(%)	Training
Lattice W_b	67.5	**61.1**	10.1	22.4	6.4	
Lattice W_1	67.1	**61.7**	11.9	20.9	5.4	CRF
	64.2	58.7	12.9	22.9	5.5	Perceptron
	67.0	59.7	11.0	22.0	7.3	Av. Perceptron

of the LIMSI decoder would allow a better match of the training and test conditions and may result in larger improvements. We refer to this here as a future work plan.

5.7.4 Discussion of the results

Some further analysis can be made concerning the improvement expected when the phoneme confusion model is introduced. It could be beneficial to retrain the acoustic model after applying the confusion model (or with the new pronunciation dictionary generated using the confusion model), which is however time-consuming and was not done in this work. Another element of the set-up that brings some difficulties to the picture is the baseline dictionary used. It is a semi-manually generated dictionary corrected to correspond to the needs of an ASR system and it already includes some of the most common variants. What is more, some of its variants were added manually and are not always consistent and may even influence negatively the training procedure. Plus, it is surely a more difficult baseline to improve than a dictionary without any variants including just the canonical pronunciation forms.

In any case, we should not forget that confusability remains a severe problem of pronunciation modeling and it is difficult to moderate it without taking into account any context. It is interesting to note that when adding to the recognition dictionary, for example, just one pronunciation variant generated by the Moses-g2p converter (see Section 3.5), the WER already degrades even though the variants are generated taking a lot of context into account. This shows again the importance of the training of the pronunciation weights. The fact that our unigram phoneme confusion model with trained weights does not add further confusability to the system, is a promising result for the use of the proposed training techniques.

Some more reasons explaining the difficulty of the task we are facing, can be searched in the characteristics of our data. We are dealing with continuous speech of very long sentences that makes the computational cost very high for training. Moreover, because of this computational cost we are obliged for the moment to work at the phoneme level and not expand the model at the sentence level. One more reason for the high computational cost is the fact that we are working with lattices and not rescoring n-best lists as in previous works. To the best of our knowledge it is the first time such a discriminative method to train pronunciation weights is applied to such a large ASR system (continuous speech with sentences of such length, baseline dictionary of 80,000 word entries). But of course in such a large system it becomes more difficult to see improvement, as well as to find the sources of errors.

5.8 Conclusion

We close this chapter by presenting some highlights of this work. First of all, using a unigram confusion model does not introduce additional confusability to the system once its weights are trained. This is an indication that the weights are trained in the right direction and further improvements may be expected if more context is added to the confusion model. Another result that is worth to highlight is the improvement over the baseline of the FST-decoder, at the phoneme and at the word level, when the confusion model is used in a post-processing step. However, integrating it in a non-FST decoder stays an open problem.

An interesting characteristic of our model is that it does not add a particular number of pronunciations per word, but instead expands the phonemic search space where pronunciation variants can be found. This correlates better with the observation that some words (for example frequent words) have many variants while others are more stable phonemically (Antilla, 2002). In addition, the phonemic analysis of the confusions shows a good representation by our confusion model of frequent phonemic variations found in real speech (Section 5.6).

It should be also noted that all experiments were conducted in a continuous speech large ASR system with long segmented sentences which is admittedly a difficult baseline. Plus, English is a language known for the difficulty it presents on the phonemic variation modelisation. As already mentioned, in our knowledge it is the first time such a discriminative method to train pronunciation weights is applied to an ASR system with long segmented sentences, as such a task can present computational difficulties, as well as difficulties in analyzing the errors and in observing some improvement.

Concerning the training of our model, two discriminative training methods were explored. Discriminative training methods are found to better control confusability as they take into account not only the correct transcription but also competing hypotheses obtained by decoding the training corpus. In addition, in this work we present a set-up and methods that can be generalized to more complicated features (i.e., heuristics, linguistic-oriented features,...(Tang et al., 2012)) without any significant changes needed. Moreover, during the training of the pronunciation weights, we make use of acoustic scores which allows to take into account some phonemic information provided by the acoustic model. Last but not least, we propose a purely data-driven method, and linguistic or heuristic initialization of the model is not needed. Plus, the pure FST implementation of the decoder and of the discriminative training enables the integration of the training modules and of the trained confusion model in any FST-based ASR system.

Working at the phoneme level also presents some advantages. It is easily generalisable as it allows us to find pronunciations also for words not present in the training data. Furthermore, if we later add the phonemic context to our confusion model, the FST-based decoding set-up allows us to deal not only with within-word but also with cross-word phonemic variation.

In the future, different objective functions can be applied, such as cost-augmented CRF and large-margin methods, while also adding more context to the confusion model is judged very important. Moreover, a regularization term should be added to the loss

function of our models to allow them a better generalization to new data. It could be interesting to compare the performance of our system using different regularization terms. An idea we would like to implement is to add the entropy as a regularization term as proposed in (Grandvalet and Bengio, 2006). This allows a further combination with the work presented in Chapter 4, where the pronunciation entropy is proposed as a measure of the pronunciation confusability.

One of the problems during the evaluation of this work was a mismatch between the training and the test conditions when our confusion model was introduced as a post-processing step to the LIMSI word recognizer (for example the acoustic scores are used in training but are not available for testing). To circumvent this, we could do the training on the phonemic transcription of the word output of the LIMSI decoder, keeping the one-best output or an n-best list or lattice of hypotheses. Moreover, the scoring procedure could be better aligned with the one followed by the LIMSI system if some normalization rules were applied on the output data and if word boundaries information was kept during the inverse decoding.

Last but not least, the proposed methods can be easily trained up to the word level keeping the same phoneme-based features but doing a word decoding instead of a phoneme one. This allows to add more complicated features, for example word-specific features for a phoneme transformation. In this way, a global control of the resulting pronunciations can be conducted. Moreover, correcting common errors in some frequently appearing words may improve a lot the accuracy.

Chapter 6

Discriminatively trained phoneme confusion model for KWS

In the previous chapter, a discriminative trained phoneme confusion model was used to expand the phonemic search space of pronunciations in ASR decoding. This approach allowed us to enrich the recognition lexicon with phonemic variation without adding undesirable confusability to the system. In this chapter, we propose the use of discriminative training to construct a phoneme confusion model, which expands the phonemic index of a KWS system. The objective function that is optimized this time is the Figure of Merit (FOM), which is directly related to the KWS performance. A phonemic index in KWS allows us to handle the problem of OOVs, but often encounters the low quality of the available phoneme recognizer. The introduced phoneme confusion model will act as a corrector of the errors of the phoneme recognizer used for the construction of the index. It can also be seen as a way to add alternative pronunciations to the query terms. These pronunciation variants will be adapted to a particular data set, as in the previous chapter. The experiments conducted on English data sets show some improvement on the FOM and are promising for the use of such technique.

6.1 Introduction

The last years there is an increasing interest towards KWS in audio data, an application that is applied as a post-processing step to ASR recognition. As the amount of real-world spoken data rapidly increases, the ability to search it efficiently for particular words or phrases of interest gains more importance. KWS aims at searching audio data and detecting any given keyword, which is typically a single word or a short phrase. KWS systems build for off-line data processing usually operate in two phases: indexing and search. The system processes the audio data once, during the indexing phase, without knowledge of the query terms. This phase is done off-line and is the more time-consuming one. The output index is stored and accessed during the search phase, in order to locate the terms and link them to the original audio (for more details see Section 2.2.5).

Word-level indexing may seem to be a straightforward solution, but it cannot handle OOV terms which are often named entities not covered by the automatic speech recogni-

tion (ASR) dictionaries. The OOV rate may increase with time, as dictionaries are usually fixed while the content of real-world data changes dynamically. Generating pronunciations for OOVs implies having a letter-to-sound system, which is often not accurate, especially for languages with limited resources. Augmenting the recognition system with pronunciation variants can help, but implies regenerating the index and may introduce confusability to the KWS system and increase the false alarms, especially if their weights are not properly tuned. Moreover, there are many applications where the performance of the word recognizer is severely degraded due to challenging audio conditions, and even in-vocabulary words are not successfully represented in the index. For these reasons, subword-level and particularly phonemic-based KWS systems have been used in the past (Saraclar and Sproat, 2004), which do not impose any vocabulary restrictions.

In the current work, a phoneme index is created from lattices generated by a phoneme recognizer, which allows us to preserve information about phonemic uncertainty of the phoneme recognition in the index. The phonemic index allows detecting any word without constraining the system to the in-vocabulary terms. However, the quality of the phoneme recognizer can be quite low, especially when dealing with data recorded under noisy or mismatched conditions. For this reason, a phoneme confusion model is introduced in this work. Its goal is to predict the deviation of the phoneme recognition output compared to the true spoken phonemes. Once applied on the index, this confusion model acts as a corrector of the recognition errors.

Our confusion model expands the index with alternative phoneme sequences, which inherently introduce additional detections when searching queries. This can be beneficial, especially if the space limitations forces us to significantly prune the phoneme lattices to keep the index to a reasonable size. However, KWS performance optimization is a counterbalancing procedure of increasing the true detections while keeping the false alarms low. To address this problem, we train the parameters of the confusion model discriminatively, where the optimized objective function is the Figure of Merit (FOM), a well-established evaluation metric of KWS performance. The FOM was also used in (Wallace et al., 2011) to directly optimize the weights of the index, which had the form of a matrix of probabilistic acoustic scores. In (Wang et al., 2009), it was used to optimize an interpolation factor when alternative pronunciations were added for OOVs. To our knowledge this is the first time it is used to train the weights of a phoneme confusion model for KWS.

The reader is considered to be familiar with the basic concepts of the theory of finite state transducers for reading the following sections. For more details, he is referred to Section 2.3.

6.2 Keyword spotting system

6.2.1 Indexing and searching representation

As already mentioned, our KWS system operates in two phases: indexing and search. In our work, the query terms are phoneme sequences and the index is constructed from a set of phoneme lattices generated for each input utterance using a phoneme recognizer. As proposed in (Can and Saraclar, 2011), the index is represented as a weighted finite

state transducer (WFST) allowing very efficient search for queries. The index WFST is constructed in such a way that every subsequence of phonemes found in any path of any phoneme lattice is represented by exactly one successful path. For such path, the sequence of input symbols corresponds to the subsequence of phonemes and the output sequence identifies the lattice/utterance. The index WFST is built on the lexicographic semiring so that each path weight is a triple representing the start time, the end time and the log posterior probability of the phoneme subsequence appearing in the lattice at this time. An example of the index can be seen in Figure 6.1. For more details on constructing the index WFST, we refer the reader to (Can and Saraclar, 2011).

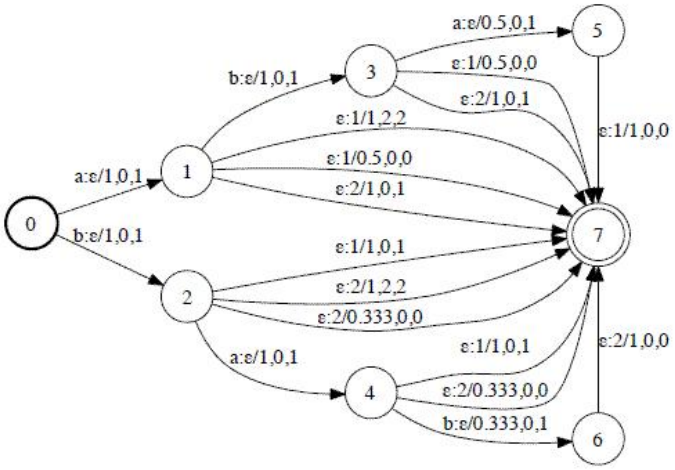

Figure 6.1: *Timed Factored Index*

The query terms are represented as weighted finite state acceptors (WFSA). In the simple case, where the query is a single phoneme sequence, the acceptor is a linear sequence of arcs with input symbols representing the corresponding phonemes. The acceptor can have, however, more complicated topology, where its individual paths represent a set of query terms we want to search for. The acceptor can also represent multiple pronunciation variants of a query word. In this last case, the weight of each path represents the (log) probability of the corresponding pronunciation variant.

Let the index and the query be represented by transducer I and acceptor Q, respectively. The search can be performed by composing the two automata $Q \circ I$ and sorting the paths through the resulting transducer with the shortest-path algorithm. Again, just like in the case of the original index WFST, each path through the composed transducer encodes information about the phoneme sequence, the lattice/utterance it was detected in and the timing. The composed transducer, however, contains only the phoneme sequences represented by Q. All the FST manipulations are realized using the OpenFst libraries (Allauzen et al., 2007).

6.2.2 Confusion model

Now, we want to take into account the fact that the query phoneme sequence can get assigned wrong posterior probability in the index (e.g. because of a systematic error of the phoneme recognizer) or is completely missing from the index (e.g. because of lattice pruning needed to obtain an index of a reasonable size). For these reasons, a confusion model is introduced reflecting our assumption that a phoneme sequence can get (with a certain probability) misrecognized as a different sequence, which we may wish to search for instead. In this work, we consider only simple context independent confusion model, assuming that each phoneme can get with a certain probability inserted, deleted or substituted by another phoneme. The confusion model is represented as a WFST on the tropical semiring with all arcs looping in a single state. For each arc, the weight $w(i, o)$ can be interpreted as the log probability that the phoneme represented by the input symbol i gets misrecognized as the output symbol o. An empty symbol ϵ can be used as input symbol to represent insertion or output symbol to represent deletion.

A query transducer Q can be now composed with a given confusion model C to obtain an expanded query transducer $Q \circ C$. For each path through $Q \circ C$, the output symbols represent the phoneme sequence we want to search for in the index instead of the original query (input symbol sequence). The weight of the path then represents the assumed probability that the output sequence (if found in the index) is in fact the misrecognized input sequence. Finally, the search of the expanded query in the index I can be performed by composing all three transducers $Q \circ C \circ I$. Alternatively, this composition can be seen as searching the original query Q in the expanded/corrected index $C \circ I$. Again, each path through the final transducer $Q \circ C \circ I$ can be interpreted as the detection of a query phoneme sequence in a particular lattice/utterance at a particular time. The weight of the k-th such path (e.g. as obtained from shortest-path algorithm) defines the detection score s_k. Note that the score combines the weight w_k of the confusion model C and the weight (log posterior) n_k of the index I as

$$s_k = w_k + n_k, \tag{6.1}$$

which naturally expresses that the final score is the product (sum in log domain) of two probabilities: 1) probability of misrecognizing the original query as a different phoneme sequence and 2) probability of detecting that sequence in the index. We can further decompose the confusion model's path weight into the contributions of the individual phonemes j as

$$w_k = \sum_j w(i_{kj}, o_{kj}). \tag{6.2}$$

It should be noted that the confusion model's output symbols o_{kj} are not visible in the final transducer $Q \circ C \circ I$, as these are consumed by the composition of C and I. However, for the discriminative training, as will be detailed below, it is still needed to keep track of all the contributing scores $w(i_{kj}, o_{kj})$ for each detection.

6.2.3 Confusion model initialization

The weights of the confusion model transducer $w(i, o)$ are the discriminatively trained parameters as will be described in Section 6.3.2. To get reasonable initial values for the weights, we obtain the one-best phoneme recognition output from our training corpora, align it with the reference phoneme sequence, and count the number of phoneme specific insertions, deletions and substitutions. The weights are thus initialized as follows: For the insertion of the output phoneme o, the weight (log probability) is set as

$$w(\epsilon, o) = \log \frac{\text{\# of insertions of phoneme } o}{\text{\# of all recognized phonemes}}. \tag{6.3}$$

All other weights are initialized as

$$w(i, o) = \log \left(P(\text{non-ins}) \frac{\text{\# of } i \text{ recognized as } o}{\text{\# of } i \text{ in the reference}} \right), \tag{6.4}$$

where the probability of the output symbol not being inserted

$$P(\text{non-ins}) = 1 - \frac{\text{\# of all insertions}}{\text{\# of all recognized phonemes}} \tag{6.5}$$

and where $o = \epsilon$ corresponds to phoneme deletion.

6.3 Confusion model training

6.3.1 The Figure of Merit

As the criterion for the discriminative training of the confusion model's parameters we use the FOM. The FOM is defined as the detection rate averaged over the range of 0 to 10 false alarms per hour and over the individual queries (Rohlicek et al., 1989). Equivalently, it can be interpreted as the normalized area under the Receiver Operating Characteristic (ROC) curve in that false alarm range. To evaluate FOM for a given data set and a given set of queries, we enumerated detections by applying the shortest-path algorithm to $Q \circ C \circ I$ transducer. Based on the reference transcription, the detections are assigned into two sets: set of true detections R^+ and set of false alarms R^-. In accordance with the FOM definition, only the top scoring detections are assigned to the sets R^+ and R^- corresponding to 10 false alarms per hour and query term. More precisely the top scoring detections are selected, so that $|R^-| \approx A = 10|Q|T$, where $|Q|$ is the number of query terms and T is the number of hours of speech. The FOM is defined as

$$FOM = \frac{1}{A} \sum_{j \in R^-} \sum_{k \in R^+} h_k H(s_k, s_j), \tag{6.6}$$

where s_k is the score for detection k as defined by equation (6.1), $h_k = 1/(|Q|Occ(k))$, $Occ(k)$ is the number of true occurrences of the query term corresponding to detection k

in the reference transcript, and

$$H(s_k, s_j) = \begin{cases} 1 & s_k > s_j \\ 0 & otherwise. \end{cases} \tag{6.7}$$

In accordance with the FOM definition, this formula can be interpreted as a numerical integration over the ROC curve: Let the outer sum, performed over false detections j, be sorted by decreasing scores s_k. Then, the sum can be seen as an integration over the false alarm rate axis (i.e. each detection j corresponds to one more false alarm) with an appropriate integration step $1/A$ corresponding to the axis scale. Therefore, each j represents a certain false alarm rate obtained with the score threshold s_k. For a given false alarm rate j, the inner sum can be interpreted as the corresponding detection rate (averaged over queries) calculated as the appropriately normalized number of true detections exceeding the threshold s_k.

As can be seen from the formula, the FOM can be also interpreted as a metric testing the discriminative power of the detection scores by expressing the KWS as the problem of ranking hits above false alarms.

6.3.2 Discriminatively optimizing the Figure of Merit

Since the FOM is not a continuous differentiable function, which is required for our optimization, we closely approximate it as

$$f = \frac{1}{A} \sum_{k \in \mathbb{R}^+} h_k \sum_{j \in \mathbb{R}^-} \varsigma(s_k, s_j) \tag{6.8}$$

$$\varsigma(s_k, s_j) = \frac{1}{1 + exp(-\alpha(s_k - s_j))}, \tag{6.9}$$

where the step function $H(s_k, s_j)$ is approximated by a sigmoid $\varsigma(s_k, s_j)$ (Raykar et al., 2007). The tunable parameter a is set to $a = 1$ for this work. Note also that, compared to equation (6.6), we have switched the order of the two sums to allow a more efficient evaluation.

The confusion model parameters are trained to maximize the objective function f on the training data and using the training queries described in Section 6.4. For the optimization, we use a simple gradient descent algorithm with a fixed step, which requires the evaluation of the derivatives

$$\begin{aligned} \frac{\partial f}{\partial w(i,o)} &= \frac{\alpha}{A} \sum_{k \in \mathbb{R}^+} h_k \sum_{j \in \mathbb{R}^-} \varsigma(s_k, s_j)(1 - \varsigma(s_k, s_j)) \\ &\quad \left[\frac{\partial s_k}{\partial w(i,o)} - \frac{\partial s_j}{\partial w(i,o)} \right], \end{aligned} \tag{6.10}$$

where $\partial s_k / \partial w(i,o)$ is simply the number of times the weight $w(i,o)$ occurs in the sum of equation (6.2).

Intuitively, in order to increase the FOM, the weights $w(i, o)$ should change such that the scores of hits increase with respect to the scores of false alarms. Note that when optimizing the weights we do not put any constraint on them, so that they may not correspond to any probabilistic model in the end.

6.4 Experimental setup

Experiments on English data were conducted. For the phoneme recognizer, acoustic models developed for the RT-04 evaluation, which were also used for the SRI STD-06 submission for the broadcast news task (Vergyri et al., 2007), were applied. In particular, gender independent cross-word triphone PLP models were trained for a set of 45 phones (including pause, reject and two hesitation specific phones). 13 PLP coefficients plus 1st, 2nd and 3rd order derivatives were used, while cepstral mean and variance normalization, vocal tract length normalization and HLDA to reduce dimentionality to 39 were applied. Decision tree state clustering was used to cluster the triphone states to about 2500 states, and 200 Gaussians per state were trained using 5 MLE training iterations and 4 discriminative (alternating MPE-MMIE) training iterations. The training data were LDC distributions including: Hub4 1996 and 1997 (200h), TDT4 (275h) TDT2 (272h) and BNr1234(2300h). The output lattices were pruned in a preprocessing step before the index construction.

The data set for the discriminative training of the confusion model's weights consisted of 8 hours of broadcast news (BN) data. The KWS performance of the proposed technique was evaluated on the NIST STD06 evaluation (NIST, 2006) set consisting of 3 hours of data. Two disjoint sets of query terms were used for training and evaluation. From the training data, 1217 query words were extracted and translated to their phonemic transcriptions. The transcriptions longer than 3 phonemes were kept. To avoid biased results, the 100 most frequent words but also the words that occurred less than 5 times were not selected. The evaluation queries were the same 1104 terms that were used for the NIST STD06 evaluation.

6.5 Results

The performance metric adopted in this work is the FOM (see Section 6.3.1), which is the one we try to optimize. The results will be presented in terms of the ROC curves showing the system performance for different operation points.

The baseline system is the one, where the KWS search is realized on the index before applying the confusion model and expanding the search space. The initialized confusion model is then applied and discriminatively trained to maximize the FOM on the training data. Figure 6.2 presents the ROC curves on the training data before training, after 10 iterations and after 50 iterations of the training algorithm. It can be seen that the area under the ROC curve is indeed increased.

We have found that it is beneficial to smooth the trained confusion model before applying it to the evaluation data. The smoothing chosen here is the linear interpolation of the

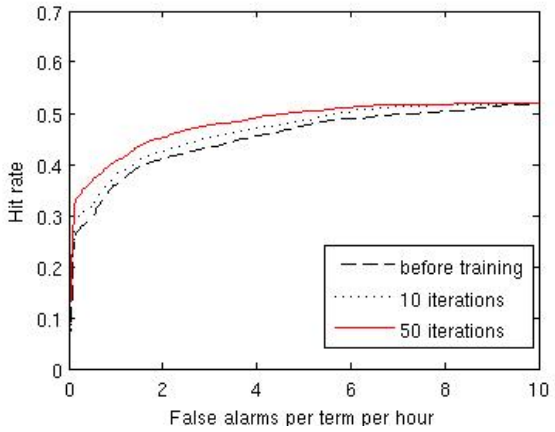

Figure 6.2: *ROC curves on training data*

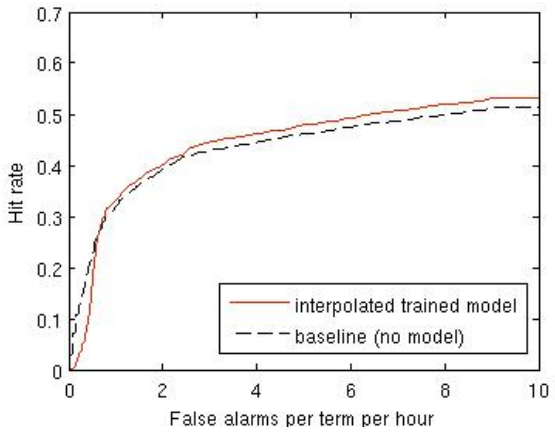

Figure 6.3: *ROC curves on evaluation data*

trained confusion model weights with a null model, which considers zero probabilities for substitutions, deletions and insertions and probabilities one for correct phoneme recognitions. The interpolation factor was heuristically set to 0.5. The curve for the smoothed trained model in Figure 6.3 corresponds to this interpolated model. Some degradation is seen for the low false alarms region (less than 1 false alarm per hour per query term), but then some improvement is observed, which increases for high false alarms numbers. For this area, if a random horizontal line is drawn in Figure 6.3, it can be actually seen that there is achieved an important decrease in the false alarms number for the same hit rate when the smoothed trained model is applied. For example, for a hit rate of 0.45, the false alarms number decreases from 5 (baseline curve) to 2.5 (interpolated trained curve) false alarms per hour and per query term. It should be noted that the value of the baseline performance is fairly acceptable for a phonemic KWS system. A last comment can be made on a slight deterioration of the KWS performance observed on the evaluation set in comparison to the training set. The KWS task seems to be more difficult on the evaluation data, which can be due to the different query term list used.

6.6 Conclusion

We have presented a phoneme confusion model for the KWS that enables recovery from recognition errors, and detection of OOVs. A discriminative approach for training its weights was applied based on the direct optimization of the FOM. The approach was tested for English. However, it is language-independent and could be applied to other languages, potentially including languages with limited resources where the OOV problem is more extensive. In terms of FOM performance a promising improvement was observed on the evaluation set.

The confusion model is applied on the index constructed using the output lattices of a phone-loop recognizer. In the future we plan to apply it also to hybrid systems that use both word and phoneme recognition. The confusion model used in this work does not take into account any phoneme context. It is our aim to try to use at least bigram phoneme confusion models and expect to achieve better KWS results. Our aim is also a better initialization of the confusion model and we have already started working in this direction. In addition, other more complex methods to train the parameters of our model could be investigated during the FOM optimization. Last but not least, currently the confusion model just add bias to the posterior scores. Instead, more complicated confusion models could be developed that operate directly on the acoustic scores from which the posteriors are computed. In this case, the confusion model could represent a multiplicative or additive correction to the acoustic scores.

Chapter 7

Conclusion and Perspectives

We close this thesis with a summary of the main findings and contributions of this work. After this, some perspectives for continuation of the current work are also discussed.

7.1 Thesis summary

The first part of this thesis was devoted to the automatic generation of pronunciations for OOVs and of pronunciation variants for the baseforms of a recognition dictionary. Some innovating SMT-inspired approaches were proposed and state-of-the-art g2p results were achieved over a difficult baseline. Then, the expanded lexicon was tested in speech recognition experiments and some improvements were noticed over a single pronunciation dictionary baseline. However, adding a lot of variants resulted in a degradation of the ASR performance. This highlighted the well-known problem of phonemic confusability when phonemic variation is added to an ASR system without any constraints. Our interest then turned in the direction of having a better understanding of these confusability phenomena and finding a way to measure and counterbalance them.

Next, pronunciation entropy was defined, a measure of the confusability introduced by the recognition lexicon in the decoding process. Experiments were conducted in order to observe how this measure is influenced when automatically generated variants are added to the lexicon. We also measured the influence of using frequency counts as weights to the pronunciations of the lexicon in contrast to no weights at all. We did not manage to find a clear correlation of this measure with the error rate of the system though.

The use of frequency counts is a very simplistic way to assign weights to pronunciations and it is restricted to words that occur in the training set. A more suitable way of choosing pronunciations and training their weights might improve the ASR performance. In this thesis, discriminative training was proposed to train a phoneme confusion model that expands the search space of pronunciations during ASR decoding. The proposed methods offer phonemic variation while keeping the confusability of the system low. Moreover, the additional variation is adapted to a particular data set and not static as in the g2p conversion task. An FST-based training and decoding was implemented and an improvement over the FST-based decoder was observed. It is not straight-forward

however to integrate our confusion model in a non-FST based decoder.

Last but not least, we expanded the discriminative training to the KWS task adopting a new objective function directly related to the KWS performance. There has been growing interest in the KWS task as the amount of available data exponentially augments and an efficient way of searching them becomes indispensable in order to be able to make the best (or any) use of them. In this work, gains were observed over the baseline when using a discriminatively trained phoneme confusion model to expand the index of a phoneme-based system.

7.2 Perspectives

We feel that most of the perspectives of this work turn around the construction of a speech-adapted pronunciation lexicon. This was actually the path this thesis took as we were understanding more about the problems and needs of pronunciation modeling nowadays. A static dictionary can give very good accuracy or recall results in g2p evaluation without managing to improve the performance of an ASR system. Data specific training of the recognition lexicon (using speech data) is preferable, as is traditionally done for the others parts of an ASR system. It is our belief that data-based lexicon adaptation will become more and more important as we move away from the standard BN recognition framework, which is more or less characterized by speakers that follow the baseform pronunciation rules. There is an increasing interest in the recognition of speech which is more spontaneous or more varied due to accents that the static dictionaries often fail to model correctly.

To construct a speech-adapted pronunciation lexicon, we should first find the right variants and then weight them such as to give them discriminative power compared to others. This is what we do with the proposed discriminative training framework using phoneme confusion pairs to capture the pronunciation variation. Concerning these phonemic features, adding more context is a promising research direction. It could also be beneficial to experiment with features providing information from other sources, for example integrating prosodic or syntactic context.

For the discriminative training proposed in this thesis, we plan to conduct experiments with objective functions in which the cost is directly integrated. It may also be interesting to try to add the pronunciation entropy to the objective function, which provides some additional information that may permit a better control of the confusability. An FST-based framework is already put in place that allows these transformations without a lot of additional programming and computational cost. One of our goals is also to move up the training to the word level keeping the features of phoneme confusions but plausibly adding word-based features (i.e., a phoneme confusion pair often occurring in a particular word). This will better account for common errors of the system which are difficult to correct otherwise. The same models could be used in other tasks without any particular changes if suitable data were available, for examples in orthographic correction.

It could also be interesting to combine the g2p conversion and the speech-adaptation approaches proposed in this thesis for the lexicon construction. More in particular, in a first place we could use the proposed g2p converter (Chapter 3) to generate alternative

pronunciations for the words of the recognition lexicon. Then, the discriminative speech-adaptation approach could be used in order to provide weights for these pronunciation variants and do some pruning on them.

For the KWS task, there is the possibility to integrate the proposed confusion model in a state-of-the-art hybrid system. In this way more gains might be seen since the proposed method would be focused to OOV words which are the ones that very often provoke errors. Moreover, for both ASR decoding and KWS, we would like to test the proposed models under different experimental conditions, better controlled and targeted to a specific problem. For example, we might see larger improvements working with accented speech, where our confusion model could capture, for example, some common accent-specific phoneme substitutions or deletions.

Appendix A

Phoneme set for American English

In the following table the 45 phoneme set used to construct the LIMSI lexicon and the acoustic models is presented. For clarity, the corresponding IPA phoneme symbols are also given with a word example of the pronunciation of each phoneme.

Table A.1: *Phoneme set for American English*

	LIMSI symbols	IPA symbols	Examples
Vowels	i	i	b<u>ee</u>t
	I	I	b<u>i</u>t
	e	eɪ	b<u>ai</u>t
	E	e	b<u>e</u>t
	@	æ	b<u>a</u>t
	∧	∧	b<u>u</u>t
	a	ɑ	b<u>o</u>tt
	c	ɔ	b<u>ough</u>t
	o	oʊ	b<u>oa</u>t
	u	u	b<u>oo</u>t
	U	ʊ	b<u>oo</u>k
	R	ɜr	b<u>ir</u>d
Diphthongs	Y	aɪ	b<u>i</u>te
	O	ɔɪ	b<u>oy</u>
	W	aʊ	b<u>ou</u>t
Fricatives	s	s	<u>s</u>ue
	z	z	<u>z</u>oo
	S	ʃ	<u>sh</u>oe
	Z	ʒ	mea<u>s</u>ure
	f	f	<u>f</u>an
	v	v	<u>v</u>an
	T	θ	<u>th</u>in
	D	ð	<u>th</u>at

Affricates	C	tʃ	cheap
	J	dʒ	jeep
Semivowels	l	l	led
	r	r	red
	w	w	wed
	y	j	yet
Plosives	p	p	pet
	t	t	tat
	k	k	cat
	b	b	bet
	d	d	debt
	g	g	get
Nasals	m	m	met
	n	n	net
	G	ŋ	thing
Reduced Vowels	x	ə	about
	|	ɪ	rating
	X	eɚ	butter
Syllabics	L		bottle
	M		bottom
	N		button
	h	h	hat
	.		silence

Appendix B

Publications

2013 **Discriminative training of a phoneme confusion model for a dynamic lexicon in ASR**
P. Karanasou, F. Yvon, T. Lavergne and L. Lamel, Proc. of Interspeech 2013

2012 **Discriminatively trained phoneme confusion model for keyword spotting**
P. Karanasou, L. Burget, D. Vergyri, M. Akbacak and A. Mandal, Proc. of Interspeech 2012

2011 **Pronunciation Variants Generation Using SMT-inspired Approaches**
P. Karanasou and L. Lamel, Proc. of ICASSP 2011, pp.4908–4911

Measuring the confusability of pronunciations in speech recognition
P. Karanasou, F. Yvon and L. Lamel, FSMNLP 2011, ACL Anthology, pp.107–115

Automatic Generation of a Pronunciation Dictionary with Rich Variation Coverage Using SMT Methods
P. Karanasou and L. Lamel, CICLing 2011, LNCS 6609, pp.506–517

2010 **Comparing SMT Methods for Automatic Generation of Pronunciation Variants**
P. Karanasou and L. Lamel, IceTAL 2010, LNCS/LNAI 6233, pp.167–178

Bibliography

M. Adda-Decker and L Lamel. Pronunciation variants across system configuration, language and speaking style. *Speech Communication*, 29:83–98, 1999.

L. Adde, B. Rveil, j.-P. Martens, and T. Svendsen. A minimum classification error approach to pronunciation variation modeling of non-native proper names. In *Proc. of Interspeech*, pages 2282–2285, 2010.

M. Akbacak, D. Vergyri, and A. Stolcke. Open-vocabulary spoken term detection using graphone-based hybrid recognition systems. In *ICASSP*, pages 5240–5243, 2008.

Y. Akita and T. Kawahara. Generalized statistical modeling of pronunciation variations using variable-length phone context. In *ICASSP*, pages 689–692, 2005.

C. Allauzen, M. Mohri, and B. Roark. Generalized algorithms for constructing statistical language models. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 40–47. Association for Computational Linguistics, 2003.

C. Allauzen, M. Mohri, and M. Saraclar. General indexation of weighted automata:application to spoken utterance retrieval. In *Proc. of HLT-NAACL*, 2004.

C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. Openfst: a general and efficient weighted finite-state transducer library. In *Proc. of the 12th international conference on Implementation and application of automata*, CIAA'07, pages 11–23. Springer-Verlag, 2007.

I. Amdal, F. Korkmazsdiy, and A.C. Surendran. Data-driven pronunciation modelling for non-native speakers using association strength between phones. In *Proc. of ASRU*, pages 85–90, 2000.

A. Antilla. *Variation and phonological theory*. Chambers, J. K., Trudgill, P. & Schilling-Estes N., Oxford:Blackwell, 2002.

X. L. Aubert. A brief overview of decoding techniques for large vocabulary continuous speech recognition. In *Automatic Speech Recognition: Challenges for the new Millenium (ASR2000)*, pages 91–97, 2000.

I. Badr, I McGraw, and J. Glass. Learning new word pronunciations from spoken examples. In *Proc. of Interspeech*, 2010.

C. Bannard and C. Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proc. of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604. Association for Computational Linguistics, 2005.

F. Beaufays, A. Sankar, S. Williams, and M. Weintraub. Learning linguistically valid pronunciations from acoustic data. In *Proc. of Interspeech*, volume 3, 2003.

F. Béchet and F. Yvon. Les noms propres en traitement automatique de la parole. *Traitement Automatique des Langues (T.A.L)*, 41(3):671–707, 2000.

J. R. Bellegarda. Unsupervised, language-independent grapheme-to-phoneme conversion by latent analogy. *Speech Communication*, 46(2):140–152, 2005.

J. Bilmes and G. Zweig. The graphical models toolkit: An open source software system for speech and time-series processing. In *ICASSP*, pages 3916–3919, 2002.

M. Bisani and H. Ney. Investigations on joint-multigram models for grapheme-to-phoneme conversion. In *Proc. of ICSLP*, 2002.

M. Bisani and H. Ney. Open vocabulary speech recognition with flat hybrid models. In *Proc. of Interspeech*, pages 725–728, 2005.

N. Bodenstab and M. Fanty. Multi-pass pronunciation adaptation. In *ICASSP*, pages 865–868, 2007.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proc. of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187. Springer, 2010.

L. Breiman, J. Friedman, C. J. Stone, and R. Olshen. *Classification and regression trees*. Chapman & Hall, 1984.

P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19 (2):263–311, 1993.

D. Can and M. Saraclar. Lattice indexing for spoken term detection. *IEEE Transactions on Audio, Speech and Language Processing*, 19(8):2338–2347, 2011.

D. Can, E. Cooper, A. Sethy, C. White, B. Ramabhadran, and M. Saraclar. Effect of pronunciations on oov queries in spoken term detection. In *ICASSP*, pages 3957–3960, 2009.

L. Chase. *Error-responsive feedback mechanisms for speech recognizers*. PhD thesis, Carnegie Melon University, 1997.

N.F. Chen. Informative dialect recognition using context-dependent pronunciation modeling. In *ICASSP*, pages 4396–4399, 2011.

S. F. Chen. Conditional and joint models for grapheme-to-phoneme conversion. In *Proc. of Eurospeech*, pages 2033–2036, 2003.

Y.-N. Chen, P. Liu, J.-L. You, and F. K. Soong. Discriminative training for improving letter-to-sound conversion performance. In *ICASSP*, pages 4649–4652, 2008.

N. Chomsky and M. Halle. *The Sound Pattern of English*. Harper & Row, New York, 1968.

G.F. Choueiter, S. Seneff, and J.R. Glass. Automatic lexical pronunciations generation and update. In *Proc. of ASRU*, pages 225–230, 2007.

A.W. Coetzee and S. Kawahara. Frequency biases in phonological variation. *Natural Language & Linguistic Theory*, pages 1–43, 2012.

M. Collins. Discriminatively training methods for hmms. theory and experiments with perceptron algorithm. In *Proc. of ACL-02:EMNLP*, volume 10, pages 1–8, 2002.

C. Cortes, M. Mohri, A. Rastogi, and M. D. Riley. Efficient computation of the relative entropy of probabilistic automata. In *Proc. of the 7th Latin American conference on Theoretical Informatics*, LATIN'06, pages 323–336. Springer-Verlag, 2006.

T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

N. Cremelie and J.-P. Martens. In search of better pronunciation models for speech recognition. *Speech Communication*, 29(2-4):115–136, 1999.

M. Dedina and H. Nusbaum. Pronounce: a program for pronunciation by analogy. *Computer Speech and Language*, 5:55–64, 1991.

S. Deligne, F. Yvon, and F. Bimbot. Variable-length sequence matching for phonetic transcription using joint multigrams. In *Proc. of Eurospeech*, 1995.

Y. Deng, M. Mahajan, and A. Acero. Estimating speech recognition error rate without acoustic test data. In *Proc. of Eurospeech*, pages 929–932, 2003.

T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence*, 1995.

M. Divay and A.-J. Vitale. Algorithms for grapheme-phoneme translation for english and french: Applications for database searches and speech synthesis. *Computational linguistics*, 23(4):495–523, 1997.

J. Eisner. Expectation semiring: Flexible em for learning finite-state transducers. In *Proc. of FSMNLP*, 2001.

J. Eisner. Parameter estimation for probabilistic finite-state transducers. In *Annual meeting-association for computational linguistics*, pages 1–8. Association for Computational Linguistics, 2002.

E. Fosler-Lussier. *Dynamic Pronunciation Models for Automatic Speech Recognition*. PhD thesis, University of California, Berkeley, 1999.

E. Fosler-Lussier and G. Williams. Not just what, but also when: Guided automatic pronunciation modeling for broadcast news. In *DARPA Broadcast News Workshop*, pages 171–174, 1999.

E. Fosler-Lussier, I. Amdal, and H.-K. J. Kuo. On the road to improved lexical confusability metrics. In *Workshop on Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology, PMLA*, pages 53–58, 2002.

E. Fosler-Lussier, I. Amdal, and H. K. J. Kuo. A framework for predicting speech recognition errors. *Speech Communication issue on Pronunciation Modeling and Lexicon Adaptation*, 46(2):153–170, 2005.

Y. Freund and R. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

T. Fukada, T. Yoshimura, and Y. Sagisaka. Automatic generation of multiple pronunciations based on neural networks. *Speech communication*, 27(1):63–73, 1999.

M. Gales and S. Young. The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304, 2007.

J. L. Gauvain, L. Lamel, and G. Adda. Partitioning and transcription of broadcast news data. In *Proc. of ICSLP*, 1998.

J.L. Gauvain, L. Lamel, and G. Adda. The limsi broadcast news transcription system. *Speech Communication*, 37(1):89–108, 2002.

M. Gerosa and M. Federico. Coping with out-of-vocabulary words:open versus huge vocabulary asr. In *ICASSP*, 2009.

K. Gimpel and N.A. Smith. Softmax-margin crfs: Training log-linear models with cost functions. In *Proc. of HLT-NAACL*, pages 733–736, 2010.

N. Goel, M. Thomas, S. Agarwal, P. Akyazi, L. Burget, K. Feng, A. Ghoshal, O. Glembek, M. Karafit, D. Povey, A. Rastrow, R. C. Rose, and P. Schwarz. Approaches to automatic lexicon learning with limited training examples. In *ICASSP*, pages 5094–5097, 2010.

V. Goel, S. Kumar, and W. Byrne. Segmental minimum bayes-risk decoding for automatic speech recognition. *IEEE Transactions on Speech and Audio Processing*, 12:234–249, 2004.

S. Goronzy and R. Kompe. Generating non-native pronunciation variants for lexicon adaptation. *Speech Communication*, 42:109–123, 2004.

Y. Grandvalet and Y. Bengio. Entropy regularization. In *Semi-Supervised Learning*, pages 151–168. MIT Press, 2006.

S. Greenberg, S. Chang, and J. Hollenback. An introduction to the diagnostic evaluation of the switchboard-corpus automatic speech recognition systems. In *Proc. of NIST Speech Transcription Workshop*, pages 16–19, 2000.

T. Hain. Implicit modelling of pronunciation variation in automatic speech recognition. *Speech Communication*, 46:171–188, 2005.

T. J. Hazen, I. Lee Hetherington, H. Shu, and K. Livescu. Pronunciation modeling using a finite-state transducer representation. *Speech Communication*, 46:189–203, 2005.

G. Heigold. *A Log-Linear Discriminative Modeling Framework for Speech Recognition*. PhD thesis, Aachen, 2010.

J. Holmes and W. Holmes. *Speech Synthesis and Recognition*. Taylor & Francis, Inc., Bristol, PA, USA, 2002.

T. Holter and T. Svendsen. Maximum likelihood modelling of pronunciation variation. *Speech Communication*, 29:177–191, 1999.

C. Huang, T. Cahen, and E. Chang. Accent issues in large vocabulary continuous speech recognition (lvcsr). *International Journal of Speech Technology*, 7:141–153, 2004.

I. Illina, D. Fohr, and D. Jouvet. Grapheme-to-phoneme conversion using conditional random fields. In *Proc. of Interspeech*, pages 2313–2316, 2011.

M. Jansche. *Inference of string mappings for language technology*. PhD thesis, OSU Linguistics Department, 2003.

F. Jelinek. A fast sequential decoding algorithm using a stack. *IBM J Research and Dev*, 13(6):675–685, 1969.

S. Jiampojamarn, C. Cherry, and G. Kondrak. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proc. of ACL-08:HLT*, pages 905–913, 2008.

D. Jurafsky, W. Ward, Z. Banping, K. Herold, Y. Xiuyang, and Z. Sen. What kind of pronunciation variation is hard for triphones to model? In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 577–580, 2001.

P. Jyothi and E. Fosler-Lussier. A comparison of audio-free speech recognition error prediction methods. In *Proc. of Interspeech*, pages 1211–1214, 2009.

P. Jyothi, E. Fosler-Lussier, and K. Livescu. Discriminatively learning factorized finite state pronunciation models from dynamic bayesian networks. In *Proc. of Interspeech*, 2012.

E. M. Kaisse. *Word-Formation and Phonology*, volume 64. Springer Netherlands, 2005.

P. Karanasou and L. Lamel. Pronunciation variants generation using smt-inspired approaches. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, pages 4908–4911, 2011.

J.M. Kessens, M. Wester, and H. Strik. Improving the performance of a dutch csr by modeling within-word and cross-word pronunciation variation. *Speech Communication*, 29: 193–207, 1999.

A. Kipp, M.-B. Weswnick, and F. Schiel. Pronunciation modeling applied to automatic segmentation of spontaneous speech. In *Proc. of Eurospeech*, pages 1023–1026, 1997.

P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics, 2003.

P. Koehn, Hoang H., A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, Cowan B., W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses:open source toolkit for statistical machine translation. In *Annual meeting-association for computational linguistics*, pages 177–180. Association for Computational Linguistics, 2007.

P. Ladefoged. *A course in phonetics (5th edition)*. Thomson Wadsworth, Boston, MA, USA, 2006.

J. Lafferty, A. McCallum, and P. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, 2001.

L. Lamel and G. Adda. On designing pronunciation lexicons for large vocabulary, continuous speech recognition. In *Proc. of ICSLP*, pages 6–9, 1996.

A. Laurent, P. Deleglise, and S. Meignier. Grapheme to phoneme conversion using an smt system. In *Proc. of Interspeech*, 2009.

T. Lavergne, O. Cappé, and F. Yvon. Practical very large scale crfs. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513. Association for Computational Linguistics, 2010.

K.F. Lee and H.W. Hon. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(11):1641–1648, 1989.

P. Lehnen, S. Hahn, A. Guta, and H. Ney. Incorporating alignments into conditional random fields for grapheme to phoneme conversion. In *ICASSP*, pages 4916–4919, 2011.

B. Logan, P. Moreno, J.-M. V. Thong, and E. Whittake. An experimental study of an audio indexing system for the web. In *Proc. of ICSLP*, pages 676–679, 2000.

L. Mangu, E. Brill, and A. Stolcke. Finding consensus among words: Lattice-based word error minimization. In *Proc. of Eurospeech*, 1999.

L. Mangu, E. Brill, and A. Stolcke. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer Speech Language*, 14(4):373–400, 2000.

D. McAllaster, L. Gillick, F. Scattone, and M. Newman. Fabricating conversational speech data with acoustic models: a program to examine model-data mismatch. In *ICSLP*, 1998.

I. McGraw, I. Badr, and J.R. Glass. Learning lexicons from speech using a pronunciation mixture model. *IEEE Transactions on audio, speech and language processing*, 21(2): 357–366, 2013.

M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

M. Mohri. Weighted automata algorithms. *Handbook of weighted automata*, 3:213–254, 2009.

M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

N. Moreau, H.G. Kim, and T. Sikora. Phonetic confusion based document expansion for spoken document retrieval. *ICSLP Interspeech 2004*, 2004.

S. Nakamura, R. Gruhn, and Harald Binder. Recognition of non-native speech using dynamic phoneme lattice processing. In *Acoustic Society of Japan, spring meeting 2002*, 2002.

NIST. The spoken term detection (std) 2006 evaluation plan, 2006. URL `http://www.nist.gov/speech/tests/std/docs/std06-evalplan-v10.pdf`.

F. J. Och and H. Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 440–447. Association for Computational Linguistics, 2000.

B. Oshika, V. Zue, R. Weeks, H. Neu, and J. Aurbach. The role of phonological rules in speech understanding research. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):104–112, 1975.

V. Pagel, K. Lenzo, , and A. W. Black. Letter-to-sound rules for accented lexicon compression. In *Proc. of ICSLP*, pages 2015–2018, 1998.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

C. Parada, A. Sethy, and B. Ramabhadran. Balancing false alrams and hits in spoken term detection. In *ICASSP*, pages 5286–5289, 2010.

D.B. Paul. Algorithms for an optimal a* search and linearizing the search in the stack decoder. In *ICASSP*, pages 693–696, 1991.

A. Paz. *Introduction to probabilistic automata*. Academic Press, Inc., Orlando, FL, USA, 1971.

F.C.N. Pereira and M.D. Riley. Speech recognition by composition of weighted finite automata. In *Finite-State Language Processing*, pages 431–453. MIT Press, 1996.

J. Pinto, A. Lovitt, and H. Hermansky. Exploiting phoneme similarities in hybrid hmm-ann keyword spotting. In *Proc. of Interspeech*, volume 4, pages 1817–1820, 2007.

D. Povey. *Discriminative Training for Large Vocabulary Speech Recognition*. PhD thesis, Cambridge University Engineering Dept, 2003.

D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah. Boosted mmi for model and feature-space discriminative training. In *ICASSP*, pages 4057–4060, 2008.

A. Prince and P. Smolensky. *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell Publishers, 2004.

H. Printz and P. Olsen. Theory and practice of acoustic confusability. In *Proc. of ISCA ITRW ASR*, pages 77–84, 2000.

M. Pucher, A. Türk, J. Ajmera, and N. Fecher. Phonetic distance measures for speech recognition vocabulary and grammar optimization. In *3rd Congress of the Alps Adria Acoustics Association*, pages 2–5, 2007.

T. Rama, A. K. Singh, and S. Kolachina. Modeling letter-to-phoneme conversion as a phrase based statistical machine translation problem with Minimum Error Rate training. In *Proc. of HLT-NAACL: Student Research Workshop & Doctoral Consortium*, 2009.

V. Raykar, R. Duraiswami, and B. Krishnapuram. A fast algorithm for learning large scale preference relations. In *Proc. of the Eleventh International Conference on Artificial Intelligence and Statistics*, pages 385–392, 2007.

G. Riccardi, R. Pieraccini, and E. Bocchieri. Stochastic automata for language modeling. *Computer Speech & Language*, 10(4):265–293, 1996.

M. Riley, W. Byrne, M. Finke, S. Khudanpur, A. Ljolje, J. McDonough, H. Nock, M. Saraclar, C. Wooters, and G. Zavaliagkos. Stochastic pronunciation modelling from hand-labelled phonetic corpora. *Speech Communication*, 29(2-4):209–224, 1999.

B. Roark, M. Saraclar, M. Collins, and M. Johnson. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proc. of the 42nd Annual Meeting on Association for Computational Linguistics*, page 47. Association for Computational Linguistics, 2004.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish. Continuous hidden markov modeling for speaker-independent word spotting. In *ICASSP*, pages 627–630, 1989.

A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer-Verlag, New York, 1978.

M. Saraclar and S. Khudanpur. Pronunciation ambiguity vs. pronunciation variability in speech recognition. In *Proc. of Eurospeech*, pages 515–518, 1999.

M. Saraclar and S. Khudanpur. Pronunciation change in conversational speech and its implications for automatic speech recognition. *Computer Speech and Language*, 18 (4):375–395, 2004.

M. Saraclar and R. Sproat. Lattice-based search for spoken utterance retrieval. *Proc. of the HLT-NAACL*, pages 129–136, 2004.

T. Sejnowski and C. Rosenberg. Nettalk: a parallel network that learns to read aloud. In *Report JHU/EECS-86/01*, 1986.

H. Shu and I. Lee Hetherington. Em training of finite-state transducers and its application to pronunciation modeling. In *Proc. of ICSLP*, pages 1293–1296, 2002.

T. Sloboda and A. Waibel. Dictionary learning for spontaneous speech recognition. In *Proc. of ICSLP*, pages 2328–2331, 1996.

N. A. Smith. *Linguistic Structure Prediction*. Graeme Hirst, University ofToronto, 2011.

M. F. Spiegel. Using the orator synthesizer for a public reverse-directory service:design, lessons, and recommendations. In *Proc. of Eurospeech*, pages 1897–1900, 1993.

V. Steinbiss. Sentence-hypotheses generation in a continuous-speech recognition system. In *Proc. of European Conference on Speech Communication and Technology*, pages 51–54, 1989.

A. Stolcke. Srilm-an extensible language modeling toolkit. In *Proc. of ICSLP*, 2002.

H. Strik and C. Cucchiarini. Modeling pronunciation variation for asr: A survey of the literature. *Speech Communication*, 29:225–246, 1999.

N. Stroppa. *Analogy-Based Models for Natural Language Learning*. PhD thesis, Tlcom ParisTech, 2005.

T. Svendsen, F.K. Soong, and H. Purnhagen. Optimizing baseforms for hmm-based speech recognition. In *Proc. of Eurospeech*, page 1, 1995.

H. Tang, J. Keshet, and K. Livescu. Discriminative pronunciation modeling: a large-margin, feature-rich approach. In *Proc. of ACL*, pages 194–203, 2012.

P. Taylor. Hidden markov models for grapheme to phoneme conversion. In *Proc. of Interspeech*, 2005.

J. Tejedor, D. Wang, S. King, J. Frankel, and J. Cols. A posterior probability-based system hybridisation and combination for spoken term detection. In *Proc. of Interspeech*, pages 2131–2134, 2009.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

M. Tsai, F. Chou, and L. Lee. Improved pronunciation modeling by inverse word frequency and pronunciation entropy. In *Proc. of ASRU*, pages 53–56, 2001.

M.-Y. Tsai, F.-C. Chou, and Lee L.-S. Pronunciation modeling with reduced confusion for mandarin chinese using a three-stage framework. *IEEE Transactions on audio, speech and language processing*, 15(2):661–675, 2007.

C. Van Bael, L. Boves, H. van den Heuvel, and H. Strik. Automatic phonetic transcription of large speech corpora. *Computer Speech and Language*, 21(4):652–668, 2007.

B. van Berkel and K. De Smedt. Triphone analysis: a combined method for the correction of orthographical and typographical errors. In *Proc. of the 2nd conf. on Applied natural language processing*. Association of Computational Linguistics, 1988.

A. van den Bosch and S. Canisius. Improved morpho-phonological sequence processing with constraint satisfaction inference. In *Proc. of the 8th Meeting of the ACL Special Interest Group on Computational Phonology and Morphology at HLT-NAACL 2006*, pages 41–49, 2006.

H. van den Heuvel, J.-P. Martens, and N. Konings. G2p conversion of names. what can we do (better). In *Proc. of Interspeech*, pages 1773–1776, 2007.

H. van den Heuvel, B. Reveil, and J.-P. Martens. Pronunciation-based asr for names. In *Proc of Interspeech*, pages 2991–2994, 2009.

C.J. Van Rijsbergen. Information retrieval. In *Butterworths, London, UK.*, 1979.

B. Vazirnezhad, F. Almasganj, and M. Bijankhan. A hybrid statistical model to generate pronunciation variants of words. In *Proc. of IEEE NLP-KE*, pages 106–110, 2005.

U. Venkataramani and W. Byrne. Mllr adaptation techniques for pronunciation modeling. In *Proc. of ASRU*, pages 421–424, 2001.

D. Vergyri, I. Shafran, A. Stolcke, V.R.R. Gadde, M. Akbacak, B. Roark, and W. Wang. The sri/ogi 2006 spoken term detection system. In *Proc. of Interspeech*, pages 2393–2396, 2007.

O. Vinyals, L. Deng, D. Yu, and A. Acero. Discriminative pronunciation learning using phonetic decoder and minimum-classification-error. In *ICASSP*, pages 4445–4448, 2009.

R. Wallace, B. Baker, R. Vogt, and S. Sridharan. Discriminative optimization of the figure of merit for phonetic spoken term detection. *IEEE Transactions on Audio, Speech and Language Processing*, 19(6):1677–1687, 2011.

C. Wang and P. Zhang. Optimization of spoken term detection system. *Journal of Applied Mathematics*, 2012.

D. Wang and S. King. Letter-to-sound pronunciation prediction using conditional random fields. *Signal Processing Letters, IEEE*, 18(2):122–125, 2011.

D. Wang, S. King, and J. Frankel. Stochastic pronunciation modelling for spoken term detection. In *Proc. of Interspeech*, pages 2135–2138, 2009.

W. Ward, H. Krech, X. Yu, K. Herold, G. Figgs, A. Ikeno, D. Jurafsky, and W. Byrne. Lexicon adaptation for lvcsr: Speaker idiosyncracies, non-native speakers, and pronunciation choice. In *Proc. of PMLA Workshop*, pages 83–88, 2002.

J.A. Wasser. English to phoneme translation, final version (4/15/85), 1985.

M. Weintraub, E. Fosler, C. Galles, Y.-H. Kao, S. Khudanpur, M. Saraclar, and S. Wegmann. Ws96 project report:automatic learning of word pronunciation from data. In *JHU Workshop Pronunciation Group*, 1996.

M. Wester. Pronunciation modeling for asr- knowledge-based and data-driven methods. *Computer Speech and Language*, pages 69–85, 2003.

G. Williams and S. Renals. Confidence measures for evaluating pronunciation models, 1998.

M. Wolff, M. Eichner, and R. Hoffmann. Automatic learning and optimization of pronunciation dictionaries. In *ISCA Tutorial and Research Workshop (ITRW) on Adaptation Methods for Speech Recognition*, 2001.

M. Wolff, M. Eichner, and R. Hoffmann. Measuring the quality of pronunciation dictionaries. In *Proc. of PMLA*, pages 117–122, 2002.

Q. Yang, J.-P. Martens, P.-J. Ghesquiere, and D. Van Compernolle. Pronunciation variation modeling for asr: large improvements are possible but small ones are likely to achieve. In *Proc. of PMLA*, pages 123–128, 2002.

S. Young. A review of large-vocabulary continuous-speech. *Speech Processing Magazine, IEEE*, 13(5):45, 1996.

F. Yvon. Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks. In *Proc. of NeMLaP*, pages 218–228, 1996.

F. Yvon, P. Boula de Mareüil, C. d'Alessandro, V. Aubergé, M. Bagein, G. Bailly, F. Béchet, S. Foukia, J.-F. Goldman, E. Keller, D. D. O'Shaughnessy, V. Pagel, F. Sannier, J. Véronis, and B. Zellner. Objective evaluation of grapheme to phoneme conversion for text-to-speech synthesis in french. *Computer Speech & Language*, 12(4): 393–410, 1998.

P. Zhang, J. Shao, J. Han, Z. Liu, and Y. Yan. Keyword spotting based on phoneme confusion matrix. In *Proc. of ISCSLP*, volume 2, pages 408–419, 2006.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.