Continuous space models with neural networks in natural language processing

Hai Son Le

UNIVERSITÉ PARIS SUD
ÉCOLE DOCTORALE D'INFORMATIQUE

Thèse

pour obtenir le diplôme de

Docteur en Sciences

Spécialité: INFORMATIQUE

LE Hai Son

Continuous Space Models with Neural Networks in Natural Language Processing

Directeur: François YVON
Co-encadrant: Alexandre ALLAUZEN

préparée au Groupe TLP, LIMSI-CNRS
soutenue le 20 Décembre 2012

Jury:

Rapporteurs:	Laurent BESACIER	-	Université Joseph Fourier
	Holger SCHWENK	-	Université du Maine
Examinateurs:	Yoshua BENGIO	-	Université de Montréal
	Hermann NEY	-	RWTH Aachen
	Michèle SEBAG	-	Université Paris-Sud
Directeur:	François YVON	-	Université Paris-Sud
Co-encadrant:	Alexandre ALLAUZEN	-	Université Paris-Sud

Tình ca - Phạm Duy

Tôi yêu tiếng nước tôi từ khi mới ra đời, người ơi
Mẹ hiền ru những câu xa vời
À à ơi ! Tiếng ru muôn đời
Tiếng nước tôi ! Bốn ngàn năm ròng rã buồn vui
Khóc cười theo mệnh nước nổi trôi, nước ơi
Tiếng nước tôi ! Tiếng mẹ sinh từ lúc nằm nôi
Thoắt nghìn năm thành tiếng lòng tôi, nước ơi . . .

Cho đất nước tôi, với tiếng Việt diệu vời
Kính tặng bố Lê Xuân Cẩn, mẹ Nguyễn Thị Thành
và chị gái Lê Thị Hà Liên

Acknowledgement

My first gratitude is to Francois Yvon and Alexandre Allauzen, my thesis supervisors, who gave me the opportunity to work in the advanced research environment at LIMSI-CNRS. During four years of experience, with their constant guidance and encouragement, my personal competencies, from foreign languages to research skills, have been improving constantly and significantly. Thanks to them, my childhood dream has come true.

I would like to thank the other members of the jury: Laurent Besacier, Holger Schwenk, Yoshua Bengio, Hermann Ney and Michèle Sebag for their questions, analyses and suggestions. Special thanks to Holger Schwenk and Laurent Besacier for spending a lot of time and effort to review this thesis.

I am particularly grateful to Ilya Oparin for helping me to work and write papers in speech recognition; to Guillaume Wisniewski for helping me to write my first scientific paper.

Working at LIMSI is like living at home. A moveable feast in Paris and hourly coffee breaks each day with TLP group members will be unforgettable memories. I had the fortune of benefiting from all of them. Here are some examples. Jean-Luc Gauvain, the team leader, does not prohibit me from massively using and sometimes destroying cluster machines. Alexandre Allauzen paid my first cup of coffee at Cité Universitaire. François Yvon set up my computer for the first time. Guillaume Wisniewski helps me to always keep in mind how hard French is. Nadi Tomeh makes me understand that the life is simpler than we always think. On the contrary, Thomas Lavergne helps me to understand that everything in this world is very complex naturally. Thiago Fraga da Silva changes my football style that now becomes more and more samba. Ilya Oparin forces me to play football at my highest level. Nadège Thorez(-Oparin) falls in love with Ilya Oparin, so he is more motivated to help me in the research. Penny Karanasou comforts me with charming stories. Lê Việt Bắc gave good comments on my CV. Artem Sokolov was convincing me to stay in France, but right away left France to Germany. I also have fruitful discussions with extremely funny stories and jokes with other limsians: Souhir Gahbiche-Braham, Marco Dinarelli, Rena Nemoto, Trần Việt Anh, Li Gong, Nicolas Pécheux, Benjamin Marie, Hélène Bonneau-Maynard, Hervé Bredin, Claude Barras, Đỗ Công Thành, Marianna Apidianaki, Gilles Adda, Martine Adda-Decker, Philippe Boula de Mareüil, Lori Lamel, Jachym Kolar, Josep Maria Crego... So, many thanks for all the great moments we shared together.

I would like to thank Lương Chi Mai for introducing me to speech and natural language processing. Thanks to Đỗ Trung Tiến, a close friend, for sharing my interpretation about life. Thanks to Lionel Messi, my endless source of inspiration. I'm also indebted to B, a strange girl from yesterday. Your beautiful and rich emotional life encourages me a lot.

Finally, my biggest gratitude is undoubtedly to my parents and my sister who are always beside me, loving and supporting me.

Contents

List of Tables

List of Figures

Introduction

Current research on Natural Language Processing (NLP) aims at converting any natural form of language data into information that can be manipulated by artificial intelligent systems. Nowadays, there are many successful applications in this field, such as Machine Translation (MT), Information Extraction (IE), Search Engine (SE), Human Computer Interface (HCI) with Automatic Speech Recognition (ASR), Text to Speech (TTS)...

There is a variety of data forms that need to be processed. Among them, texts are a very important part because they implicitly encode most of our knowledge about languages. How to successfully learn from this knowledge source is a crucial question. Language modeling was brought in so as to partially answer this question. The purpose of a language model is in general to capture and to model regularities of language, thereby capturing morphological, syntactical and distributional properties of a given language. Ideally, it must be able to assess the likeliness of any sentence in a given context.

Over the last decades, due to the quick and consistent evolution of the available data for many languages, in most tasks, the statistical approach to language modeling almost swept out deterministic (grammar-based) approaches. On the one hand, statistical models can implicitly embed linguistic knowledge by handling a large quantity of data. On the other hand, formal and explicit models of syntax and semantic can only take into account a limited amount of evidence. This implies that their integration in large scale NLP applications seems to be rather ineffective when compared with statistical models. Moreover, formal linguistic models require manually annotated data. This kind of resources is very expensive, and in most of the case, their generalization to different languages or applications is quite limited. For example, Part-Of-Speech (POS) tagged texts are only available for newspaper texts and therefore, this valuable knowledge source can not readily be used in speech recognition applications. Another example is semantic networks (WordNet), which are only available for a few languages. On the contrary, statistical language models only require raw training material that can be nowadays easily harvested[1]. The most successful approaches to date are based on *n*-gram assumption and the adjustment of statistics from the training data by applying smoothing and back-off techniques, notably Kneser-Ney technique proposed in 1995 (Kneser and Ney, 1995; Chen and Goodman, 1998), some twenty years ago.

However, it should be emphasized that in spite of their prevalence, conventional *n*-gram based language models still suffer from several limitations that could be intuitively overcome by consulting human expert knowledge. One critical limitation is that, ignoring all linguistic properties, they treat each word as one discrete symbol with no relation with the others. For example, assume

[1]Of course, training texts must be carefully selected and processed, and how to design a well suited training corpus is still a complex task

the training data contains the following sentence: *"Alice is the only girl that Bob loves"*, it should help the model to capture the likeliness of a new sentence such as: *"Carole is the only woman that Bob loves"*. The reason is that *"Alice"* and *"Carole"* are very similar, likewise for the two words *"girl"* and *"woman"*. Unfortunately, conventional n-gram based language models cannot readily use this information. Another point is that, even with a huge amount of data, the *data sparsity* issue always has an important impact, so the optimal value of n in the n-gram assumption is often 4 or 5, equivalent to the implication that words only depend on the 3 or 4 previous words. Using only up to 4 previous words is insufficient in practice. To illustrate, an example can be taken from the quote of Turing (1950): *"Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."*. We see that the word *"the"* in *"the use of words"* mostly depends on *"I believe that"*. However, even 6-gram language models fail to capture this dependency.

Using word similarities in language modeling to remedy the data sparsity issue has long been shown to be helpful. Many ideas can be examined in this way. There are several tasks in NLP such as POS Tagging that try to cluster words into categories distinguished based on human analysis. Their output categories can then be served as relevant sources of information for several approaches to statistical language modeling (Jelinek, 1990; Niesler, 1997; Chelba and Jelinek, 2000; Oparin et al., 2008). In general, these approaches can be viewed as an effort to take advantage of word similarities as words in the same categories tend to exhibit the same distribution in similar contexts. But their embedded information seems to be too global because the number of categories is limited; this might be because word classes distinction are derived from human linguistic knowledge, which have many ambiguities, redundancies and are not well defined for each particular task. Other methods instead, suggest getting word classes automatically. A canonical example is the class-based approach proposed in (Brown et al., 1992). But word clusterings obtained in this way is at an expensive cost and remaining "hard" as each vocabulary word is often assigned to only one class. Furthermore, for large scale tasks, their achievements are often limited.

Recently, one of the most successful attempt that tries to directly learn word similarities is to use *distributed word representations* (Bengio, Ducharme, and Vincent, 2000) in language modeling, where distributionally words, which have semantic and syntactic similarities, are expected to be represented as neighbors in a continuous space. These representations and the associated objective function (the likelihood of the training data) are jointly learned using a multi-layer neural network architecture. In this way, word similarities are learned automatically. Moreover, unlike standard n-gram word based models, in this approach, language models can efficiently take into account longer contexts in order to mitigate the data sparsity problem. This approach has shown significant and consistent improvements when applied to ASR (Schwenk, 2007;

Emami and Mangu, 2007; Kuo et al., 2010) and Statistical Machine Translation (SMT) tasks (Schwenk, Dchelotte, and Gauvain, 2006). Therefore, continuous space language models have become increasingly used. These successes have revitalized the research on neuronal architectures for language models, and given rise to several new proposals.

A major difficulty with the continuous space neural network based approach remains the computational burden, which does not scale well to the massive corpora that are nowadays available. For this reason, the first contribution of this dissertation is the definition of a neural architecture which makes them well suited for large scale frameworks where significant improvements in both ASR and SMT are reported. Furthermore, several insightful analyses on their performances, their pros and cons, their induced word space representation are also provided. Then, the second contribution is the successful adoption of the continuous space neural network into a machine translation framework. New translation models are proposed and reported to achieve significant improvements over state-of-the-art baseline systems.

The structure of this dissertation is organized as follows. First, in Chapter 1, we briefly present the principles of language modeling along with its most successful approaches to date, some ideas about their advantages and their weaknesses in order to assess the impact and the contribution of the continuous space neural network language model approach. In Chapter 2, our new proposed architecture to continuous space neural network language models, called Structured OUtput Layer (SOUL), is proposed. In this chapter, we provide an overview of this approach, whereas Chapter 3 is dedicated to a comprehensive investigation of the SOUL architecture. Word embeddings induced from SOUL NNLMs are further analyzed in Chapter 4. In Chapter 5, we continue with a study on their possibility of taking into account long contexts. Their extension to translation models is finally investigated in Chapter 6.

Part I

Language Modeling and State-of-the-art Approaches

Chapter 1

Language Modeling

Contents

In this chapter, we describe the basic knowledge of Language Modeling along with its state-of-the-art approaches in order to draw the general context of our research. Since there exist many efforts to summarize the current research in language modeling such as the work of Rosenfeld (2000), or more experimentally, the work of Mikolov et al. (2011a), our goal here is rather to focus on analyzing the relation between approaches based on a continuous word space representation, the main subject of our study, and the other ones.

The content of this chapter is structured as follows. Section 1.1 is devoted to the definition of language models, their role in Natural Language Processing (NLP), notably in two tasks: Automatic Speech Recognition (ASR) and Statistical Machine Translation (SMT). They are the two main frameworks that we later use to conduct our experiments. Their evaluation metrics are then described in detail in Section 1.2, one derives from the information theory (perplexity) and two are defined with respect to an application (Word Error Rate for ASR and Bilingual Evaluation Understudy for SMT) . After that, we continue with the description of the most successful approaches to language modeling in Section 1.3, from the smoothing techniques to more sophisticated approaches which try to take into account information such as word classes, syntactic, semantic knowledge ... Their advantages and their drawbacks are discussed in order to analyze why it is very hard to consistently and significantly bring improvements over state-of-the-art language models which are still based on smoothing and back-off techniques that exist from twenty years ago. We finish this chapter by sketching out a recent promising approach based on a continuous word space representation in Section 1.4, from which our work directly inherits.

1.1 Introduction

Language Models (LMs) are an indispensable source of knowledge in many applications of NLP. They encompass a prior knowledge about the regularity of a text, measuring how fluent and likely it is to be spoken or written so as to lead artificial systems to generate texts that can be comprehensible by human.

A statistical language model aims at computing the likeliness of all possible word strings. Mathematically, by considering natural language as a stochastic process, it is formulated as a probability distribution $P(w_1^L)$ over sequences of words w_1^L in $\mathcal{V}^+$ where w_1^L is a shorthand for $w_1, w_2, \dots, w_L$ and $\mathcal{V}$ is a finite vocabulary. For example, it should assign a much larger probability for *"Let music be the food of love"* than *"Let music be the foot of dove"* because intuitively, the former is more likely to occur than the latter in current English.

Trying to directly estimate the probability of a whole string is not tractable. This probability is therefore usually factorized in a left-to-right manner as:

$$P(w_1^L) = P(w_1|\text{<s>}) \prod_{l=2}^{L} P(w_l|\text{<s>}w_1^{l-1}) \tag{1.1}$$

$$= \prod_{l=1}^{L} P(w_l|h_l), \tag{1.2}$$

where a token <s> (or sometimes $<w_0>$) is used to represent the beginning of the string and h_l is a history used to predict w_l, the l^{th} word. In this equation, we also consider a special token </s> as the last word w_L of w_1^L. Note that, the latter token is introduced to make the sum of probability over all possible strings equal to 1.

This factorization seems to make the task easier because now, instead of estimating one complicated distribution over word strings, we have to deal with many more, but simpler, distributions involved in predicting the next word w given each history h, $P(w|h)$. But regarding the requirement of modeling the distributions of many discrete random variables (words in a sequence), this problem remains challenging, especially in real-world NLP applications where $\mathcal{V}$ typically contains several thousands to millions of words. As a result, in the following sections, we will see that, in most approaches, some further simplifying assumptions need to be imposed.

For an ASR system, the place of the language model can be defined from the point of view of a *noisy channel* system (Shannon, 1948) where an acoustic signal (X) is considered as the output signal of a source message W being passed through a noisy channel. The goal is to recover the original message from the noisy data, or equivalently to solve the problem defined by the following formula:

$$\begin{aligned}
\hat{W} &= \underset{W}{\operatorname{argmax}}\; P(W|X) && (1.3)\\
&= \underset{W}{\operatorname{argmax}}\; \frac{P(X|W) \times P(W)}{P(X)} && (1.4)\\
&= \underset{W}{\operatorname{argmax}}\; P(X|W) \times P(W) && (1.5)
\end{aligned}$$

Here, Bayes rule is applied to decompose the original task into two different ones which involve *an acoustic model*, the first term $P(X|W)$, for the acoustic likelihood and *a language model*, the second term $P(W)$, for the prior distribution over sentences[1]. As can be seen from the previous example, due to the acoustic similarity, the two sentences *"Let music be the food of love"* and *"Let music be the foot of dove"* if viewed as candidates of an ASR system will be almost impossibly distinguished without the evidence provided from the language model.

In practice, because of having different output spaces, the scale of the language model and that of the other model cannot be compared. Consequently, we need to introduce weights to reconcile the two components of Equation (1.5):

$$\hat{W} = \underset{W}{\operatorname{argmax}}\; P(X|W)^{\lambda_{X|W}} \times P(W)^{\lambda_W} \qquad (1.6)$$

For an SMT system, in the simplest case, we follow the same scene keeping in mind that now X is used to present a source text in a foreign language that we need to translate and the first term $P(X|W)$ is given by *a translation model*. In fact, for this application, there are often more than two models to be mixed, therefore, the Bayes equation is replaced by a *log-linear* generalization of Equation (1.5):

$$\hat{W} = \underset{W}{\operatorname{argmax}} \left[\exp\left(\sum_{i=1}^{F} \lambda_i f_i(W, X) \right) \right], \qquad (1.7)$$

where f_i is used to define feature functions of W and X, each of them being associated with one model used by the system. Language models are considered as one feature function which takes only output words as arguments to

[1]In general, the term "word string" should be used instead of "sentence" but since W is often restricted to be a sentence, from now, when there is no confusion, we prefer to use "sentence".

capture the fluency of a sentence. In a well-defined system, language models often take a relatively great weight, meaning that they have a great influence on the final performance.

Motivations leading to the log-linear model are more from practical than mathematical inspiration. Models in Equation (1.6) have different degree of confidence since they are based on several assumptions which are not fully correct but necessary to make them tractable. Instead of jointly optimizing all parameters of all models, it is more straightforward to train each model separately and then use weights to arbitrate their contribution. Moreover, viewed in this way, it is easier to naturally include additional models from different approaches regardless of their information overlap. One extreme case is the work of Chiang, Knight, and Wang (2009) where about ten thousand features were used. In practice, adjusting the weights on development data always leads to better results.

To find the optimal weights, there exist several algorithms. For example, within the SMT framework, in case of using few features, we can use Minimum Error Rate Training (MERT) introduced in (Och, 2003). For a larger number of features, it seems preferable to use Margin Infused Relaxed Algorithm (MIRA) from (Watanabe, Asahara, and Matsumoto, 2007), its refinement in (Chiang, Marton, and Resnik, 2008; Chiang, Knight, and Wang, 2009) or recently, its batch-mode presented in (Cherry and Foster, 2012). Another alternative is to use Pair Wise Ranking Optimization (PRO) introduced in (Hopkins and May, 2011).

1.2 Evaluation Metrics

In this section, we present three ways to assess the performance of language models. The first one is a system-independent method based on information theory. Following this approach, it requires only a test text that is taken from the same source of the training data. The other ones are actually two metrics dedicated to specific applications, namely Automatic Speech Recognition (ASR) and Statistical Machine Translation (SMT). Following these more practical approaches, we can evaluate the direct contribution of language models to finalized tasks, which gives a clearer view of their actual quality.

1.2.1 Perplexity

This method uses *perplexity* as a metric. Basically, it assesses the quality of a language model based on the average likelihood of unseen data D according to the model:

$$\text{perplexity}(D, P) = \exp\left(-\frac{\sum_{l=1}^{L} \log P(w_l|h_l)}{L}\right), \tag{1.8}$$

where L is used to denote the number of words in the unseen data D and $P(w_l|h_l)$ is given by the language model. The exponential term is in fact the negative of the average likelihood. It can be interpreted as the empirical estimate cross entropy between the unknown distribution $\bar{P}(w|h)$ from which the test data is drawn and the language model:

$$\text{cross\_entropy}(\bar{P}, P) = -\sum_{w,h} \bar{P}(w,h) \log P(w|h) \tag{1.9}$$

Equation (1.8) can also be rewritten in order to interpret perplexity as a geometric average (branching factor) of predictions given by this language model:

$$\text{perplexity}(D, P) = \prod_{l=1}^{L} P(w_l|h_l)^{-\frac{1}{L}} \tag{1.10}$$

A lower perplexity implies smaller geometric average, corresponding to the fact that the language model fits (or predicts) better the data. One advantage is that its computation is straightforward and efficient without access to any application. For this reason, perplexity is usually used for both tuning and shallow evaluations. It also has theoretically elegant properties as its logarithm is an upper bound on the number of bits per word expected in compressing a text and actually used in predicting the performance of text compression methods.

On the downside, perplexity depends not only on the language model but also on the data D and on the underlying vocabulary. So comparisons are only meaningful when the same data and the same vocabulary are used. Furthermore, the correlation between the perplexity and the performance of most tasks is not obvious. Observed improvements of perplexity do not necessarily lead to better application results. In (Rosenfeld, 2000), the author claimed that in an ASR system, a 5% perplexity reduction is not significant at all, whereas $10 - 20\%$ is noteworthy and more than 30% is significant. Another weakness is that perplexity cannot be used to evaluate un-normalized language model[2] that can evidently be useful in the application point of view, for example, a stupid back-off model (Brants et al., 2007). For these reasons, there have been some attempts trying to find other metrics that remain task-independent but are more correlated with application results. For example, in (Chen, Beeferman, and Rosenfeld, 1998), the authors proposed another measure that is easily calculated but better predicts ASR performance than perplexity. An adaptation of the Shannon game (Shannon, 1951) to evaluate language models of different vocabularies was studied in (Bimbot et al., 2001). Another example can be taken from (Zweig and Burges, 2012) where Microsoft Research Sentence Completion Challenge for advancing language modeling was introduced as a possible new metric. Despite many efforts, perplexity hitherto remains the main metric for language model development.

[2]where the output distributions are not proper and not guaranteed to sum to one

1.2.2 Word Error Rate

For ASR applications, the straightforward way to evaluate the contribution of a language model is through Word Error Rate (WER). In order to calculate this score, we need to compare the hypothesis of the speech recognizer with the reference (the correct transcript produced by humans). WER is then defined as the edit distance between the reference and the decoder hypothesis, which is measured as the minimum number of substitutions, deletions or insertions necessary to convert one string into another:

$$\text{WER} = \frac{S + D + I}{L} 100\%, \tag{1.11}$$

where L is a total number of words in the manual transcription text, S is a total number of substitutions, D is a total number of deletions and I is a number of insertions demanded.

The first weakness is that WER is computationally expensive because it involves an entire decoding. Moreover, it is speech recognizer and task dependent. Apart from language model, there is a great number of other factors that affect speech recognition performance such as the language model weight, the word penalty, the search algorithm, the integration of the language model and the acoustic model ... restricting the comparison between language models is only meaningful when evaluated with the same system or in contrastive experiments.

1.2.3 Bilingual Evaluation Understudy

For machine translation applications, there are several evaluation metrics similar to WER. But unlike in ASR tasks where the use of WER is dominant, there are a lot of metrics proposed for SMT tasks such as BLEU, TER, METEOR, NIST ... but the evaluation problem remains so that: *"More has been written about MT evaluation over the past 50 years than about MT itself"* (Lopez, 2008). For this reason, MT evaluation remains an important concern, as illustrated for instance by the shared task in Workshop on Statistical Machine Translation (WMT). Most metrics show some correlations with human judgment but they are never satisfying, especially when dealing with systems from very different approaches. Within the scope of this dissertation, Bilingual Evaluation Understudy (BLEU) is always used as it is often considered as a criterion to tune and to evaluate translation systems and has proved to work quite well with statistical based systems.

Introduced by (Papineni et al., 2002), BLEU measures the similarity of n-gram count vectors of the reference translations and the candidate translation predicted by a translation system. n-grams of different orders are counted and then interpolated. The typical maximum order of n-gram is 4. Because BLEU is based on precision and the formulation of recall is not trivial over multiple references, brevity penalty (BP) is introduced to lower the strong bias towards short sentences. BP is calculated as follows:

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ \exp(1 - \frac{r}{c}) & \text{otherwise} \end{cases}, \tag{1.12}$$

where r is the effective length of the reference (calculated as the sum of the single reference which is closest to the hypothesis translation) and c is the length of hypothesis. If we use p_i and λ_i to denote the precision score of i-grams in the test corpus and its associated weight respectively, BLEU can then be computed as:

$$\mathrm{BP} \times \exp\left(\sum_{i=1}^{n} \lambda_i \log p_i\right) \tag{1.13}$$

In practice, all weights are equal to $\frac{1}{n}$. As BLEU is a precision measure, higher values of BLEU indicate better results.

For a sake of clarify, here is an example in the case of one sentence taken from the original article on BLEU. Suppose that we have 3 references:

1. It is a guide to action that ensures that the military will forever heed Party commands
2. It is the guiding principle which guarantees the military forces always being under the command of the Party
3. It is the practical guide for the army always to heed the directions of the party

and 2 hypotheses:

1. **It is** to insure **the** troops **forever** hearing **the** activity guidebook **that party** direct
2. **It is a guide to action which ensures that the military always** obeys **the command of the party**

From the point of view of human judgments, the second hypothesis is certainly better. It is also preferred according to BLEU metric as it has more identical n-grams (in bold), they are longer and furthermore, its length is closer to those of the references.

Though BLEU has frequently been reported as having a good correlation with human judgments, the use of BLEU is recommended to be restricted from "tracking, broad, incremental changes to a single system, comparing systems which employ similar translation strategies" (Callison-Burch, Osborne, and Koehn, 2006). In this paper, BLEU scores were shown not to correspond to the scores of human evaluations: The system which was considered as the best by human was ranked at the 6th by BLEU. Finally, it is worth noticing that BLEU reported here are from systems following the same phrased based approach, therefore, escaped from this issue. Final conclusions drawn from BLEU scores often go in the same line as those from other usual evaluation metrics for machine translation.

1.3 State-of-the-art Language Models

Many approaches have been proposed over the last decades, most of them trying to maximize the log-likelihood (analog to minimize the perplexity) of the training data with some constraints and smoothing techniques in order to guarantee the generalization to the new unseen data, i.e., to prevent overfitting. We are going to first examine the most widely used n-gram language models. An n-gram model relies on a Markovian assumption that each word depends only on $n-1$ previous words:

$$P(w_1^L) \approx \prod_{l=1}^{L} P(w_l|w_{l-n+1}^{l-1}), \tag{1.14}$$

letting negative and null indices for w_i represent a start symbol (<s>) to simplify the notations.

It is natural to use $n-1$ to denote the order of language models because it is the order of a Markov model reflected by this equation. Unfortunately, in the literature, authors always use n instead, so do we hereafter.

For a better view, the general form of the conditional probability, $P(w|h)$, is modified to become $P(w_n|w_1^{n-1})$, where $w = w_n$ stands for the predicted word and $h = w_1^{n-1}$ stands for its history comprising the $n-1$ previous words. To estimate this probability, we can simply use the number of occurrences of sequences of words (n-grams) in some training data. Let $c(w_1^n)$ denote the number of times that this n-gram occurs, estimated probability can be derived as follows:

$$P_{\text{MLE}}(w_n|w_1^{n-1}) = \frac{c(w_1^n)}{c(w_1^{n-1})} \tag{1.15}$$

This is the maximum likelihood estimate (MLE) of the n-gram probability on the training data. Note that, if during testing, we meet a history which is not seen in the training data, meaning that the denominator of the equation is null, we are obliged to use the MLE of a smaller order.

Despite the use of the n-gram assumption, MLE remains unreliable and tends to underestimate the probability of very rare n-grams, which are hardly observed even in huge corpora. It is a direct consequence of the Zipf's law stating that the frequency of any word is inversely proportional to its rank in the frequency table (Zipf, 1932). Moreover, it assigns null probability to a great number of plausible n-grams which is problematic in most applications. As a result, several smoothing techniques were proposed to address these issues. Some of them will be briefly described in the next sections.

1.3.1 Smoothing Techniques

Conventional smoothing techniques, such as Kneser-Ney and Witten-Bell, use lower order distributions to provide a more reliable estimate, especially for the probability of rare n-grams. (Chen and Goodman, 1998) provides an empirical overview and (Teh, 2006) gives a Bayesian interpretation for these methods. These techniques aim at smoothing the MLE distributions in two steps: (i) discounting a probability mass for observed n-grams; (ii) redistributing this mass to unseen events. Generally, distributions are adjusted in such a way that small probabilities are increased and high probabilities are diminished to prevent zero probabilities and improve the model prediction accuracy. They are divided into two types: *interpolated* and *back-off* that differ crucially in the way of determining the probabilities for observed n-grams: While the former uses lower order distributions, the latter does not.

According to (Kneser and Ney, 1995), back-off smoothing techniques can be described using the following equation:

$$P(w_n|w_1^{n-1}) = \begin{cases} \alpha(w_n|w_1^{n-1}) & \text{if } c(w_1^n) > 0 \\ \gamma(w_1^{n-1})\beta(w_n|w_2^{n-1}) & \text{otherwise} \end{cases} \tag{1.16}$$

So if an n-gram occurs at least one time in the training data, we simply use $\alpha(w_n|w_1^{n-1})$ while in the other case, we use the information from its lower order gram $\beta(w_n|w_2^{n-1})$ with $\gamma(w_1^{n-1})$, a normalization factor. $\alpha(w_n|w_1^{n-1})$ is expected to not contain any information of lower order distributions.

On the other hand, interpolated smoothing techniques can be represented using the following equation:

$$P(w_n|w_1^{n-1}) = \kappa(w_1^{n-1})P_{\text{MLE}}(w_n|w_1^{n-1}) + (1-\kappa(w_1^{n-1}))P(w_n|w_2^{n-1}), \tag{1.17}$$

where $\kappa(w_1^{n-1}) \in [0,1]$ is an interpolation weight depending on each n-gram w_1^{n-1}.

Note that if we set:

$$\gamma(w_1^{n-1}) = 1 - \kappa(w_1^{n-1}), \tag{1.18}$$

and then:

$$\alpha(w_n|w_1^{n-1}) = \kappa(w_1^{n-1})P_{\text{MLE}}(w_n|w_1^{n-1}) + \gamma(w_1^{n-1})P(w_n|w_2^{n-1}), \tag{1.19}$$

we will see that interpolated models can be described in a similar way for back-off models as in Equation (1.16). The difference is that the computation of $\alpha(w_n|w^{n-1})$, in this case, needs to invoke lower order information $P(w_n|w_2^{n-1})$.

In general, a back-off equation (1.16) can represent both types of language models based on back-off and interpolated smoothing techniques. As a consequence, the term *back-off language model* is often used as a general name for conventional n-gram language models with smoothing.

1.3.1.1 Absolute discounting

Absolute discounting introduced in (Ney, Essen, and Kneser, 1994) can be considered as an interpolated smoothing technique where the higher-order distribution is created by discounting each nonzero count by a constant amount, $0 \leq \mathrm{dc} \leq 1$. Its formulation is similar to Equation (1.17):

$$P(w_n|w_1^{n-1}) = \frac{\max\{c(w_1^n) - \mathrm{dc}, 0\}}{\sum_{w_n} c(w_1^n)} + (1 - \kappa(w_1^{n-1}))P(w_n|w_2^{n-1}) \quad (1.20)$$

To satisfy a normalization requirement, we must have:

$$\kappa(w_1^{n-1}) = 1 - \frac{\mathrm{dc}}{\sum_{w_n} c(w_1^n)} N_{1+}(w_1^{n-1}\bullet), \quad (1.21)$$

where $N_{1+}(w_1^{n-1}\bullet)$ is used to define the number of word types preceded by w_1^{n-1}:

$$N_{1+}(w_1^{n-1}\bullet) = |w_n : c(w_1^n) > 0| \quad (1.22)$$

According to the original article, dc can be set through deleted estimation on the training data to be equal to $\frac{n_1}{n_1+2n_2}$ where n_c is the number of n-gram types with exactly c occurrences. As this is a recursive equation, we need to define the 0^{th} order distribution as a uniform distribution. It implies that the first order distribution is proportional to the word frequencies in the corpus.

1.3.1.2 Interpolated Kneser-Ney smoothing

Interpolated Kneser-Ney smoothing technique (Kneser and Ney, 1995) has been widely reported to achieve state-of-the-art performance. It is an extension of absolute discounting, having the same form as Equation (1.20).

To describe it, two notations are required. First, we use $N_{1+}(\bullet w_2^n)$ to define the number of word types preceding w_2^n:

$$N_{1+}(\bullet w_2^n) = |w_1 : c(w_1^n)) > 0|, \quad (1.23)$$

and $N_{1+}(\bullet w_2^{n-1}\bullet)$ stands for the number of word type pairs that have w_2^{n-1} inbetween:

$$N_{1+}(\bullet w_2^{n-1}\bullet) = |(w_1, w_n) : c(w_1^n) > 0| = \sum_{w_n} N_{1+}(\bullet w_2^n) \quad (1.24)$$

Compared to the absolute discounting, the modification is made at the computation of the first order distribution where:

$$P(w_n) = \frac{N_{1+}(\bullet w_n)}{N_{1+}(\bullet\bullet)} \tag{1.25}$$

Using this technique, conditional probabilities can be all explicitly estimated without using the recursive equation because the previous equation can be generalized to higher orders:

$$P(w_n|w_1^{n-1}) = \frac{N_{1+}(\bullet w_2^n)}{N_{1+}(\bullet w_2^{n-1}\bullet)} \tag{1.26}$$

The main motivation leading to this approach is explained from an analysis of Chen and Goodman (1998). The authors stated that *"a lower-order distribution is a significant factor in the combined model only when few or no counts are present in the higher-order distribution. Consequently, they should be optimized to perform well in these situations"*. In particular, unigram probabilities are useful just in case we have to estimate the probabilities of an unseen bigram. To make this idea clearer, we can take an example from the same article. Suppose that in the training data, the word "Francisco" occurs much more often than the word "information". At the same time, it follows a single history "San" while the latter is preceded by a large number of different words. Now, if we need to assign the probabilities to two unseen bigrams "useful Francisco" and "useful information", that of the latter should be higher as intuitively, there are more chances that the word "information" is used after a novel history. Following the absolute discounting approach, they will be incorrectly estimated because they will be proportional to unigram probabilities, and we have assumed that the unigram probability of "Francisco" is larger. It suggests that unigram probability of a word shouldn't be proportional to the number of occurrences but instead to the number of different words observed in its history: the "count of counts" N_{1+} factors.

The *modified interpolated Kneser-Ney* smoothing technique, proposed in (Chen and Goodman, 1998), is another version where the discount (dc) depends on the number of occurrences of n-grams:

$$P(w_n|w_1^{n-1}) = \frac{c(w_1^n) - \mathrm{dc}(c(w_1^n)), 0}{\sum_{w_n} c(w_1^n)} + (1-\kappa(w_1^{n-1}))P(w_n|w_2^{n-1}), \tag{1.27}$$

where:

$$\mathrm{dc}(c) = \begin{cases} 0 & \text{if } c = 0 \\ \mathrm{dc}_1 & \text{if } c = 1 \\ \mathrm{dc}_2 & \text{if } c = 2 \\ \mathrm{dc}_{3+} & \text{if } c \geq 3 \end{cases} \tag{1.28}$$

Now, the normalization term, $1 - \kappa(w_1^{n-1})$, is computed as follows:

$$1 - \kappa(w_1^{n-1}) = \frac{\mathrm{dc}_1\, N_1(w_1^{n-1}\bullet) + \mathrm{dc}_2\, N_2(w_1^{n-1}\bullet) + \mathrm{dc}_{3+}\, N_{3+}(w_1^{n-1}\bullet)}{\sum_{w_n} c(w_1^n)}, \quad (1.29)$$

where

$$N_i(w_1^{n-1}\bullet) = |w_n : c(w_1^n) = i|, \quad (1.30)$$
$$N_{i+}(w_1^{n-1}\bullet) = |w_n : c(w_1^n) \geq i| \quad (1.31)$$

Estimates for optimal dcs can be directly computed from the statistic of the training data as follows:

$$\begin{aligned} y &= \frac{n_1}{n_1 + 2n_2}, \\ \mathrm{dc}_1 &= 1 - 2y\frac{n_2}{n_1}, \\ \mathrm{dc}_2 &= 2 - 3y\frac{n_3}{n_2}, \\ \mathrm{dc}_{3+} &= 3 - 4y\frac{n_4}{n_3}, \end{aligned} \quad (1.32)$$

where n_c is again, the number of n-grams with exactly c occurrences.

In many cases, this version is reported to perform better than the original one. Recently, an extended version with more sophisticated ways of estimating discounting parameters is proposed in (Andres-Ferrer, Sundermeyer, and Ney, 2012).

1.3.1.3 Stupid back-off

Stupid back-off introduced in (Brants et al., 2007) is motivated by the requirement to train language models on extremely large corpora (up to 2 trillion tokens). Instead of applying any discounting, it directly uses the relative frequencies:

$$\mathrm{Scr}(w_n|w_1^{n-1}) = \begin{cases} P_{\mathrm{MLE}}(w_n|w_1^{n-1}) & \text{if } c(w_1^n) > 0 \\ \gamma\, \mathrm{Scr}(w_n|w_2^{n-1}) & \text{otherwise} \end{cases}, \quad (1.33)$$

where γ is a back-off factor that is practically set to 0.4. As no normalization is done, these are not probabilities but scores. With data of such large order of magnitude, the lack of normalization does not seem to cause any degradation on the system performance. Concretely, the performance of stupid back-off models has been shown to approach that of Kneser-Ney when trained on data of more than 8 billions of tokens within a phrase-based machine translation framework. It probably demonstrates that the choice of smoothing technique has a little influence when the training data reaches billions of tokens.

In summary, even with more sophisticated smoothing techniques, the main weakness of word based n-gram language models is that they always rely on a discrete representation of the vocabulary, where each word is associated with a discrete index. In this model, the morphological, syntactic and semantic relationships which structure the lexicon are completely ignored, which has a negative impact on the generalization performance of the model. As seen later, this is the main motivation leading to the continuous space based approach for language modeling, which aims at encoding all those abstract word relationships into a unique continuous space. Apart from that, various approaches have been proposed to overcome this limitation, notably the use of word classes (Brown et al., 1992; Niesler, 1997), of explicit integration of morphological information in random forest models (Xu and Jelinek, 2004; Oparin et al., 2008), of exponential models (Lau, Rosenfeld, and Roukos, 1993; Rosenfeld, 1996; Chen, 2009b), of topic or semantic based models (Bellegarda, 1998; Gildea and Hofmann, 1999; Mrva and Woodland, 2004), of factored models (Bilmes and Kirchhoff, 2003) or of generalized back-off strategies (Bilmes et al., 1997) ... Several of these extensions, which are related to the work of this thesis, are briefly described in the next sections.

1.3.2 Class-based Language Models

As an attempt to make use of the similarity between words, class-based language models were introduced in (Brown et al., 1992) and then, deeply investigated in (Niesler, 1997). First, words are clustered into classes. The generalization is then achieved based on the assumption that the prediction for rare or unseen word n-grams can be made more accurate by taking into consideration the richer statistics of their associated (less sparse) class n-grams. In general, the mapping from words to classes can be many-to-one or many-to-many, corresponding respectively to hard and sort clusterings. Soft clustering requires a marginalization over all possible classes in the calculation of the n-gram probabilities, which is intractable in most cases. Therefore, in practice, words are often supposed further to belong to only one class.

There are two major issues in the class-based approach. The first one concerns the way of using word classes in estimating word probabilities. Let k_i be the class of word w_i, then several ways could be followed, for example, as in the original approach:

$$P(w_n|w_1^{n-1}) = P(w_n|k_n)P(k_n|k_1^{n-1}), \tag{1.34}$$

or a more complex way as in (Goodman, 2001b; Emami and Jelinek, 2005):

$$P(w_n|w_1^{n-1}) = P(w_n|k_1^n, w_1^{n-1})P(k_n|k_1^{n-1}, w_1^{n-1}) \tag{1.35}$$

For the former model structure, using maximum likelihood estimation, the first term $P(w_n|k_n)$ is the number of times that the word in class k_n is w_n in the training text. By converting words in the training text into their associated classes and then considering each class as a new word unit, the second term is estimated using conventional smoothing techniques such as Kneser-Ney smoothing. As a result, after determining a class for each word, the estimation of the two probabilities in Equation (1.34) is straightforward. For more complicated structures such as defined by Equation (1.35), more sophisticated estimations need to be employed (see (Goodman, 2001b), Section 1.3.7 or Section 1.3.8 for more details).

The other issue with this approach is how to automatically induce these classes as the quality of clustering has an important effect on final results. Concretely, it should maximize the log-likelihood of the training data. Considering a history to contain solely a previous word, this log-likelihood can be written as follows:

$$\begin{aligned}
\mathcal{L}(D_{\text{train}}) &= \sum_{w_{n-1}w_n} c(w_{n-1}w_n) \log P(w_n|w_{n-1}) \\
&= \sum_{w_{n-1}w_n} c(w_{n-1}w_n) \log P(w_n|k_n)P(k_n|k_{n-1}) \\
&= \sum_{w_{n-1}w_n} c(w_{n-1}w_n) \log \frac{c(w_n)}{c(k_n)} \frac{c(k_{n-1}k_n)}{c(k_{n-1})} \\
&= \sum_{k_{n-1}k_n} c(k_{n-1}k_n) \log c(k_{n-1}k_n) - \sum_{k_{n-1}} c(k_{n-1}) \log c(k_{n-1}) \\
&\quad - \sum_{k_n} c(k_n) \log c(k_n) + \sum_{w_n} c(w_n) \log c(w_n), \qquad (1.36)
\end{aligned}$$

where c represents the count in the training set.

As the last term does not depend on word classes, the optimal clustering is the one that maximizes the remaining summation, i.e., the average mutual information of adjacent classes. Though, finding this clustering is impractical due to high computational cost. As a result, an incremental method based on a greedy algorithm, called an *exchange algorithm* is often used. It was improved further in (Kneser and Ney, 1993). The main idea is that, at each iteration, each word in the vocabulary is moved tentatively to all the classes and finally assigned to the class that results in the highest log-likelihood measure. After several iterations, the algorithm converges to a local optimum. The resulting classes can somehow group together word with similar properties, even though it only takes bigram information into account. The extension to a larger order (trigram) was proposed in (Martin, Liermann, and Ney, 1998).

The major issue with this algorithm is it is greedy, which means that the final clustering is only locally optimal. Therefore, in (Emami and Jelinek, 2005), the authors proposed to combine models with different clusterings. They are obtained by randomizing the various design parameters: the choice of clustering data, the choice of words to be clustered, the initialization, and the number of classes. This method is very similar to a random forest based approach for language modeling as will be seen later in Section 1.3.6.

The idea of using word clustering to mitigate the data sparsity problem is helpful. However, in practice, the final results have been mixed at best. While for small settings, good improvements in perplexity in combination with other n-gram based models have been found, it has been rare that class-based language models have significantly improved results. Another disadvantage that hinders the practical use of class-based language models is that, given the quantity of data available nowadays, the computational complexity of the clustering algorithm becomes important: its learning time can be counted in days. Compared to an approach which is based on a continuous word space representation, the main subject studied in this dissertation, the word clustering of class-based language models is too severe, even in the complicated case where several random clusterings are taken into account. Furthermore, the word clustering in the class-based approach is learned separately, not directly to maximize the word probabilities. As a consequence, it is not guaranteed that an intuitively good word clustering will lead to a better performance.

1.3.3 Structured Language Models

Structured Language Models (SLMs) (Chelba and Jelinek, 2000; Roark, 2001; Filimonov and Harper, 2009) are one of the first successful attempts to introduce syntactic information into statistical language models. The main idea is to use the syntactic structure (a binary parse tree) when predicting the next word so as to filter out irrelevant history words and to focus on the important ones. In practice, as represented in Figure 1.1 in case of trigrams, to predict the last word "after", in lieu of using the two previous words "of cents" as would n-gram based LMs, SLMs use the two last *exposed headwords* (heads of phrases) according to the parse tree: "contract" and "ended" which are intuitively, stronger predictors. Compared to standard n-gram LMs, an advantage of SLMs is that they can use long distance information.

The algorithm proceeds as follows: from left to right, the syntactic structure of a sentence is incrementally built. As for each sentence, there is not a unique possible parse tree T, SLMs need to estimate the joint probability $P(w_0^L, T)$ and then induce the probability of the sentence by marginalizing out the variable T. SLMs consist of three modules:

- WORD-PREDICTOR predicts the next word given context words and their associated POS tags.
- TAGGER predicts the POS tag of the next word given this word, context words and their associated POS tags.
- CONSTRUCTOR grows the existing parse tree of context words with the next word and its POS tag.

This approach has been shown to achieve improvements both in terms of perplexity and WER for ASR systems via word lattice rescoring, over small scale baseline systems (Chelba and Jelinek, 2000). Incorporating neural network

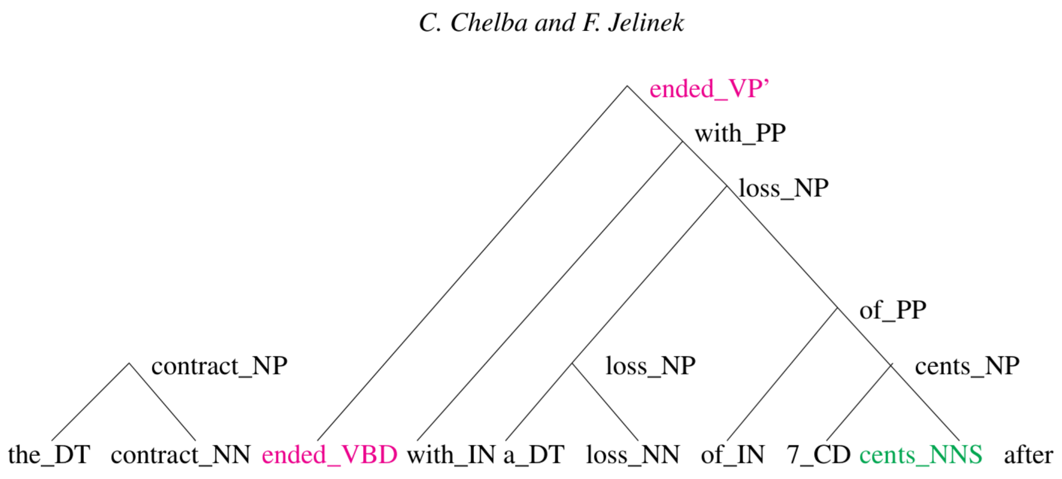


Figure 1.1: *Example for structured language models. This figure is borrowed from (Chelba and Jelinek, 2000).*

architectures into the SLM framework was investigated in (Emami and Mangu, 2007). The idea of using exposed headwords is also used in (Khudanpur and Wu, 2000) with maximum entropy models, or in neural network based models (Kuo et al., 2009). Recently, the advantage of modeling long-dependencies of SLMs over standard n-gram LMs has been again demonstrated in (Rastrow, Khudanpur, and Dredze, 2012). The local context issue is partially overcome in the continuous space neural network approach where long range dependencies are captured, but in an usual way, i.e., by increasing the model order, to at least 7 for the above example. In the neural network framework, the relevance of predictors in the context of the $n-1$ previous words can somehow be learned automatically (see Chapter 5 for more details). The major issue with SLMs is the introduction of other complex random variables (the binary parse tree) into the framework and the attempt to estimate the joint distributions that make models more complex. There are three conditional probabilistic submodels that need to be learned. For this reason, the application of SLMs in large scale tasks is usually to be too expensive.

1.3.4 Similarity based Language Models

In (Dagan, Pereira, and Lee, 1994; Dagan, Lee, and Pereira, 1999), a general method for using word similarity in language modeling was proposed. It is based on the assumption that if two words w_1 and w'_1 are similar, then w'_1 can provide information for the estimation of the probability of unseen word pairs containing w_1. Let $\text{Sim}(w_1, w'_1)$ denote a similarity measure for a word pair (w_1, w'_1), $\mathbb{N}(w_1)$ denote the neighbors, i.e., the most similar words of w_1, then the probability of a bigram is defined as follows:

$$P_{\text{sim}}(w_2|w_1) = \sum_{w'_1 \in \mathbb{N}(w_1)} \frac{\text{Sim}(w_1, w'_1)}{\sum_{w''_1} \text{Sim}(w_1, w''_1)} P(w_2|w'_1), \tag{1.37}$$

where $P(w_2|w'_1)$ is computed using any appropriate smoothing technique.

Word similarity measures can be derived from the statistics of a training corpus, using for instance, Kullback-Leibler (KL) divergence:

$$\text{Dis}(w_1||w'_1) = \sum_{w_2} P(w_2|w_1) \log \frac{P(w_2|w_1)}{P(w_2|w'_1)} \tag{1.38}$$

This term is well defined only when $P(w_2|w'_1) > 0$ whenever $P(w_2|w_1) > 0$. This assumption holds for all smoothing methods. The similarity measure is then defined as an exponential function of the KL distance:

$$\text{Sim}(w_1, w'_1) = e^{-a\,\text{Dis}(w_1||w'_1)}, \tag{1.39}$$

where β is introduced to control the relative contribution of neighbor words: with a large a, the closest words get more weight; when a decreases, distant words get more influence.

For a particular choice of Dis, we define also $\mathbb{N}(w_1)$ as the most k nearest words whose distance to w_1 is smaller than a threshold s. The hyper-parameters a, s and k are tuned experimentally.

Compared to class-based approach, they share the same intuition of using similar events to make the estimation of other events more accurate. However, while class-based methods have a good probabilistic interpretation, similarity based methods are justified only empirically. It can be considered as one attempt to use the similarity between words in a more flexible way, i.e., without the definition of class. In (Dagan, Lee, and Pereira, 1999), this approach was shown to achieve a modest improvement in a small experiment restricted to bigram probability estimation. Its extended version for longer context is possible but needs to be further investigated. For reference, , there are some recent studies on this approach such as the work of Gosme and Lepage (2011); Gillot and Cerisara (2011). As shown later, trying to take into account the similarity between words is also a main motivation of the continuous space based approach. However, their ways of considering the similarity between words are very different in two main points: continuous space neural network language models have a probabilistic interpretation and the similarity between words is learned automatically by optimizing the log-likelihood probabilities of the training data. Another advantage with the continuous space based approach is that it can consider much longer contexts.

1.3.5 Topic and semantic based Language Models

There are several attempts that try to introduce semantic information in language modeling. Among them, topic based approach which was proposed in (Gildea and Hofmann, 1999) can be taken as one salient example. In principle, topic based language models aim at handling global dependencies between words, for instance, at the document level. Suppose that the training corpus is a set of documents, topics are introduced as latent variables in a framework where the probability of a word w given a document d is decomposed as follows:

$$P(w|d) = \sum_t P(w|t)P(t|d) \tag{1.40}$$

Here t is interpreted as a topic and a d, a document defines a mixture of unigram distributions.

Suppose that we have a vocabulary containing V words and B documents. From the training corpus, we compute a co-occurrence matrix $\mathbf{C} \in \mathbb{R}^{V \times B}$ by keeping track of which word is found in what document: each element $C(i, j)$ or $C(w_i, d_j)$ is the number of times the word w_i occurs in d_j. Then, we can use an Expectation-Maximization (EM) algorithm to fit the set of parameters θ implied by $P(w|t)$ and $P(t|d)$ so as to maximize the log-likelihood of the training data:

$$\mathcal{L}(\theta, \mathbf{C}) = \sum_w \sum_d C(w, d) \log \sum_t P(w|t)P(t|d) \tag{1.41}$$

The E-step computes the expectation of latent variables as:

$$P(t|w, d) = \frac{P(w|t)P(t|d)}{\sum_{t'} P(w|t')P(t'|d)} \tag{1.42}$$

and then, The M-step adusts the parameters of the model:

$$P(w|t) = \frac{\sum_d C(w, d)P(t|w, d)}{\sum_{w'} \sum_d C(w', d)P(t|w', d)} \tag{1.43}$$

$$P(t|d) = \frac{\sum_w C(w, d)P(t|w, d)}{\sum_{t'} \sum_w C(w, d)P(t'|w, d)} \tag{1.44}$$

When testing, to compute the probability of a word w given a history h, we need to consider the history as a pseudo-document, replacing d by h in Equation (1.40). Not like in n-gram approaches, h can be used in a more general way, to represent any possible type of causal contexts, i.e., from the last $n - 1$ words as in n-gram based approaches, to the current sentence, or even to the current document. While the first term $P(w|t)$ is already available and should be reasonably kept unchanged, the second term $P(t|h)$ needs to be determined to reflect the topic distributions in current documents. A single step of an online version of the EM algorithm (suggested in (Gildea and Hofmann, 1999) and extended further in (Mrva and Woodland, 2004)) is applied:

$$P(t|h_1) = P(t) = \sum_w \sum_d C(w,d)P(t|d) \tag{1.45}$$

$$P(t|h_l) = \frac{1}{1+a} \frac{(P(w_l|t)P(t|h_{l-1})^{\text{cs}(l)}}{\sum_{t'} (P(w_l|t')P(t'|h_{l-1})^{\text{cs}(l)}} + \frac{l-1+a}{i+a} P(t|h_{l-1}), \tag{1.46}$$

where h_1 (h_l) stands for the context used to predict word w_1 (w_l).

In this equation, $\text{cs}(i)$ is the confidence score, used to take into account the reliability of words recognized from speech decoder. a is used to emphasize the prior topic distribution at the beginning (the first a words) because at this time, there is not enough information to accurately determine the topic of the document.

Because of they consider documents (or histories) as "bag of words", thus ignoring the information of word orders and syntax, topic based models should not be considered as an alternative but rather as complementary to conventional n-gram models. There are several ways of combining them. The best one seems to be a unigram rescaling method (Gildea and Hofmann, 1999) where:

$$P(w|h) \propto P_n(w|h) * \frac{P_t(w|h)}{P_u(w)}, \tag{1.47}$$

where P_n and P_t are given respectively by the conventional n-gram model and the topic based model respectively. P_u is obtained from the unigram distribution. The main idea is that we should use a unigram probability to penalize the use of the topic based model when predicting common words because they often occur with almost equal frequency in all documents, meaning that their behavior is not well captured by the topic based model.

This approach is actually an application of Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 1999) based method as described in (Mrva and Woodland, 2004). The first idea of using the document information in language modeling can be traced back to (Bellegarda, 1998) where the semantic links between words and document are captured by using Latent Semantic Analysis (LSA) (Deerwester et al., 1990). LSA is an information retrieval method that tries to learn the semantic relationships between words and documents in a corpus. Words and documents are represented as vectors in a space of large dimension. First, a word-document matrix $\mathbf{G}$ is constructed based on the statistics of the co-occurrences between words and documents (of the co-occurrence matrix $\mathbf{C}$). Then, the singular value decomposition (SVD) of the matrix is computed:

$$\mathbf{G} \approx \hat{\mathbf{G}} = \mathbf{RSB}^T \tag{1.48}$$

Word and document representations are respectively the left singular vectors of $\mathbf{G}$ (in $\mathbf{R} \in \mathbb{R}^{V\times M}$) and the right singular vectors of $\mathbf{G}$ (in $\mathbf{B} \in \mathbb{R}^{B\times M}$)

forming a space of dimension M. If two words tend to occur in the same kind of documents, they are close in this space. However, because the information encoded in the word space does not contain syntactic and word order information, the definition of word similarity is different from those of other approaches, e.g., class-based, continuous space based … Two close documents often have a similar semantic content. As a consequence, words and documents having a strong semantic relationship are also expected to be close. The similarity measures in each case are defined differently, so on for the distance. For example, the similarity Sim and the distance Dis between words w_i and w_j can be defined as follows:

$$\operatorname{Sim}(w_i, w_j) = \cos(R_{i,:}\mathbf{S}, R_{j,:}\mathbf{S}), \tag{1.49}$$

$$\operatorname{Dis}(w_i, w_j) = \arccos \operatorname{Sim}(w_i, w_j), \tag{1.50}$$

where $R_{i,:}$ and $R_{j,:}$ are the i^{th}, j^{th} line vectors of $\mathbf{R}$.

To use these similarities in inference, the history is considered again as a pseudo-document, being projected in the document space. The probabilities are then induced by normalizing the distance between words and this pseudo-document.

To summarize, topic based and LSA based approaches both focus on using semantic information. By considering a more general context, usually back to the first word of a test document to guess the topic (the domain) information of the text, they can be also considered as adaptation methods for language modeling. Little improvements in terms of perplexity were observed when combining them with standard n-gram based language models in small tasks. Unfortunately, they did not translate into improvements in real world applications. Another issue is that in many cases, the document information of a corpus is not available or ill-defined. Another topic adaptation approach for language modeling was also investigated in (Chen, Seymore, and Rosenfeld, 1998), but with the use of another model structure, an unnormalized exponential models. Recently, following the same direction, (Chien and Chueh, 2011) introduces an extension, namely, Dirichlet Class Language Models. Based on Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan, 2003; Griffiths and Steyvers, 2004), a more advanced method for document modeling, they are shown to achieve gains over the LSA based language model in a small speech recognition task. Note that, in Section 4.3, according to the results on a word relatedness task, the word space of continuous space neural network language models is shown to encode some semantic information and is significantly better than that induced from LSA methods.

1.3.6 Random Forest Language Models

Introduced for the first time in (Xu and Jelinek, 2004), Random Forest Language Models (RFLMs) are a set of Decision Tree Language Models (DTLMs) including randomization in the tree-growing algorithm. The underlying idea of using several decision trees (DTs) is that each DT constructed in this way is only locally optimal, hence; a collection of them would be closer to global optimum, i.e., generalizing better to unseen data. Combination is performed through the interpolation of all DTLMs with equal weights. The optimal number of DTs is large, from 50 to 100 trees are often needed to form a sufficiently rich forest.

A decision tree as proposed in (Breiman et al., 1984) is a kind of classifier. Its application for language modeling is first proposed in (Bahl et al., 1989) and further studied in (Potamianos and Jelinek, 1998). A DT is a tree consisting of nodes. At each node, we assign one question to partition this node into several subnodes (its children). In the context of language modeling, each node corresponds to a cluster of histories and each question is asked to divide this cluster into several subsets. At the end, the set of histories is divided into classes; each of them corresponds to a leaf of the tree. Histories of the same class (in the same leaf) share the same distribution over words. There are two major problems with this technique that need to be addressed: question design at nodes or the construction of a tree and probability smoothing technique at leaves or data fragmentation issues.

As DTLMs are often restricted to consider binary trees, at each node, the n-gram set is divided into two subsets using any possible "yes/no" question (predictor). These questions are asked about information encoded in the history, from simple word based such as: *"The first previous word is an element of the following set* $\{w_1, w_2; \ldots\}$*?"* to more complicated linguistically oriented: *"Is the POS tag for the second previous word is NOUN?"*. All n-grams which answer "yes" will proceed to one branch going out of a node and the other n-grams will follow the other branch.

The construction of deterministic DTLMs is not globally optimal since it is based on a greedy approach to reduce the uncertainty about the event to be predicted, using principles that are analog to entropy reduction. At first, all histories are pooled together to create a node called root. A number of splitting steps is then applied. At each step, a current leaf of the tree is chosen; its set of histories is divided into two subsets. This leaf becomes an internal node; two new leaves (its children) are then added to handle these new subsets. The splitting criterion is to maximize the log-likelihood of the training data, therefore, the needed information at each stage is only statistics associated with the node of interest. Ideally, the likelihood increase of all possible questions at each node should be estimated so as to be able to select the best question for each node. Expensive calculation makes it unfeasible and a greedy approach must be followed as in the class-based approach.

Suppose that the original n-gram set at one node is separated into two subsets $(S, \bar{S})$, the log-likelihood under this split can be formulated as:

$$\begin{aligned}\mathcal{L}(S) &= \sum_w \left[c(w,S) \log \frac{c(w,S)}{C(S)} + c(w,\bar{S}) \log \frac{c(w,\bar{S})}{c(\bar{S})} \right] \\ &= \sum_w \left[c(w,S) \log c(w,S) + c(w,\bar{S}) \log c(w,\bar{S}) \right] \\ &- c(S) \log c(S) - c(\bar{S}) \log c(\bar{S}),\end{aligned} \tag{1.51}$$

where $c(S)$ represents the number of occurrences of an event S observed in the training data. This quantity measures the goodness of a question locally for this node. In practice, ranking all possible questions at each node is impossible so the splitting method based on a fast *exchange algorithm* (Martin, Liermann, and Ney, 1998) (used also in class-based frameworks) is applied. In principle, the n-gram set of the node can be considered as a set of disjoint basic elements. They have one special property that we can always combine any subset of basic elements to form a question. The set of histories is initially divided into two subsets of basic elements: (itself, ∅). The best subsets are then induced by iteratively moving elements from one set into the other in such a way that the likelihood increases, until no move is found to satisfy this criterion. The gain on held-out data is finally used as a critical measure to decide whether this split is finally retained.

It is worth noticing that a conventional n-gram based language model can be interpreted as one special case of DTLM. For example, a DT with a question about one individual word at each node is a bigram model.

Even though building random forests from multiple local optimal deterministic DTLMs can potentially yield some improvements, more achievements will be expected if randomization is introduced in the construction, in order to create a more flexible structure, a randomized DT. There are two basic sources for that. Firstly, the initialization of node splitting can be accomplished randomly, i.e., using two randomize subsets $(S, \bar{S})$ instead of (itself, ∅). Secondly, predictor selection can also be made random but with care if a question with low predicting power is picked in one of the top nodes, this may result in early tree-growing stop, leading to a very shallow final tree. Therefore, the s% (to be tuned) worse questions according to the measure defined by Equation (1.51) are excluded. All good enough questions remaining after filtering have equal opportunities to be selected.

Once the construction of the tree is finished, it is possible that some n-grams of a certain leaf will have zero probability. Smoothing is therefore necessary. One can benefit from upper node distributions for smoothing, or instead, discount DT probabilities according to basic smoothing techniques such as Kneser-Ney. Another smoothing technique based on a recursive equation on the tree structure was also proposed in (Oparin, Lamel, and Gauvain, 2010b).

Due to the fact that RFLMs in theory can take many different questions, this approach is claimed to be a promising model to incorporate multiple knowledge sources. In the literature, they have been reported to achieve significant improvements when combined with conventional n-gram based models on small and medium size tasks, especially for inflectional languages such as Czech or Russian with the use of morphological features (Oparin et al., 2008). Their main weakness is that the number of nodes grows quickly with the context length, the corpus size and the number of possible questions. This method is therefore very memory and time consuming with high computational demand. For large scale tasks, several simplifications are required, resulting in smaller or even no significant achievements (Oparin, Lamel, and Gauvain, 2010b). Compared to RFLMs, continuous space based language models can reconcile several sources of information in a simpler and more flexible way as shown in Chapter 4 where their word space representation is demonstrated to efficiently group together words which have similar morphological information (lemma, inflection, POS tag . . .).

1.3.7 Exponential Language Models

An exponential language model, or equivalently maximum entropy language model, is proposed in (Lau, Rosenfeld, and Roukos, 1993; Rosenfeld, 1994). It consists of a set of feature functions $\mathbf{F} = \{f_i(w,h)\}$ and of an equal number of weights $\Lambda = \{\lambda_i\}$ where the conditional probability distributions are defined as:

$$P(w|h) = \frac{\exp(\sum_{i=1}^{F} \lambda_i f_i(w,h))}{Z(h)}, \tag{1.52}$$

where $Z(h)$ is a normalization factor that is computed as follows:

$$Z(h) = \sum_{w} \exp(\sum_{i=1}^{F} \lambda_i f_i(w,h)) \tag{1.53}$$

We can introduce in $\mathbf{F}$ arbitrary feature functions of the word-history pair (w,h). A history h is not restricted to the $n-1$ previous words as in n-gram approaches but can run from the first word of a document to the last previous word of the predicted word. Therefore, there are a lot of feature functions that can be used: usual n-grams, caching or skipping n-grams, word triggers (long distance word pairs) . . . as shown in the study of Rosenfeld (1994). Particularly, the models that take word triggers as features are often called trigger based language models.

As usual, exponential models are trained by maximizing the log-likelihood of the training data D_{train}, equivalently to maximize:

$$\sum_{w,h \in D_{\text{train}}} P_{\text{train}}(w|h) \log P(w|h) \tag{1.54}$$

In this equation, P_{train} is the empirical distribution estimated on the training data, being computed as follows:

$$P_{\text{train}}(w|h) = \frac{c(w,h)}{\sum_{w'} c(w',h)}, \tag{1.55}$$

where c stands for the count in the training data.

Calculating partial derivatives of this equation implies that exponential models have to satisfy the constraint that the expectation of each feature is equal to its empirical expectation:

$$\sum_{w,h \in D_{\text{train}}} P(w|h) f_i(w,h) = \sum_{w,h \in D_{\text{train}}} P_{\text{train}}(w|h) f_i(w,h) \tag{1.56}$$

In practice, to avoid overfitting, a regularization term is always added (Chen and Rosenfeld, 2000). The objective function was demonstrated to be convex, so the parameters Λ can straightforwardly be estimated using a gradient descent or any other iterative algorithm. However, the training procedure is often slow, because it is done through many iterations, and furthermore, at each iteration, for each example in the training data, the calculation of the normalization term Z, which involves the summation over all words in the vocabulary, is required. For the same reason, inference is also very time consuming within, for instance, an ASR system.

As summing over all words in the vocabulary is very expensive, it is very hard to train such kind of model in large scale frameworks where a usual vocabulary has at least several thousand words. In (Goodman, 2001a), the author proposed an efficient method to drastically reduce the computational time. Inspired by class-based technique, the main idea is to factorize the computation of the normalization factor into two cheaper ones. Suppose that each word w belongs to only one class k in $\mathbf{K}$, the conditional probability can be computed in a different way:

$$\begin{aligned} P(w|h) &= \sum_{k'} P(w,k'|h) \\ &= \sum_{k'} P(w|k',h)P(k'|h) \\ &= P(w|k,h)P(k|h) \end{aligned} \tag{1.57}$$

Note that the above assumption about the number of classes for words implies that $P(w|k',h) = 0$ with all k' other than k. The two resulting terms have the same form as $P(w|h)$ in Equation (1.52) and hence can be computed using the same exponential formula. Estimating their normalization factors requires to sum over all words w in class k for the first term and over all possible classes k for the second term. In general, these two numbers can be chosen to be much smaller than the vocabulary size, implying that the complexity of the proposed exponential class-based language model is much cheaper than the original one. As the continuous space language models are shown to have the same problem, in Chapter 2, based on this idea, we propose a new structure for this type of model in order to speed-up the computational time.

Despite that exponential models are based on well-defined and theoretical supports, their results in large scale frameworks are somehow limited. It may be due to the fact that most of features (usual or skip n-gram indicators) are similar to the input of conventional back-off models, so that combining both does not necessarily lead to improvements. Another problem with this approach is that when dealing with a large corpus over a large vocabulary, the use of a sufficient context size is impossible because the number of potential features (and parameters) becomes too large to handle. The function of exponential models is analogous in form to continuous space neural network models. The main difference is that the features of the former are discrete, chosen at the beginning and kept unchanged during training whereas those of the latter are continuous and learned automatically.

1.3.8 Model M

Recently, the Model Ms have been proposed and shown to achieve state-of-the-art performances (Chen, 2009b). This is a type of exponential class-based language model, but is based on a different way of using class information to shrink the model size. In (Chen, 2009a), the cross-entropy on the test data was shown to be linear in the size of the model:

$$\mathcal{H}_{\text{test}} \approx \mathcal{H}_{\text{train}} \frac{\gamma}{\text{N}_{\text{train}}} \sum_{i=1}^{F} |\lambda_i|, \tag{1.58}$$

where $\mathcal{H}$ stands for a cross-entropy, N_{train} is the number of events (n-grams) in the training data, F is the number of parameters and γ is a constant independent of domain, training set size and model type.

Therefore, by decreasing their size, one can hope to obtain models that can achieve a better generalization. This is carried out by introducing new features based on a heuristic from (Chen, 2009b): *"Identify groups of features which will tends to have similar λ_i values. For each such feature group, add a new feature to the model that is the sum of the original features"*. The set of potentially satisfying features are induced from word classes.

Replacing the general history h in Equation (1.57) by the $n-1$ previous words as Model M follows the standard n-gram based approach, we have:

$$P(w_n|w_1^{n-1}) = P(w_n|w_1^{n-1}, k_n)P(k_n|w_1^{n-1}) \tag{1.59}$$

Model M is then obtained by setting:

$$P(w_n|w_1^{n-1}, k_n) = P_e(w_n|w_1^{n-1}, k_n), \tag{1.60}$$

$$P(k_n|w_1^{n-1}) = P_e(k_n|k_1^{n-1}, w_1^{n-1}), \tag{1.61}$$

where $P_e(z|y)$ is used to denote the output of an exponential model whose feature functions are restricted to be indicator functions of any suffix of yz. For example, the features of $P_e(w_n|w_{n-1}, k_n)$ are of the form w_n, $k_n w_n$ or $w_{n-1}k_n w_n$.

The first term of Equation (1.59) is therefore estimated using an exponential model which uses features from k_n and w_1^n. The second term of Equation (1.59) is estimated using an exponential model which uses features from k_1^n and w_1^{n-1}.

Note that, in the left hand side of Equation (1.61), the use of w_1^{n-1} to represent a history is enough as it already represents all context information. On the contrary, in the right hand side, k_1^{n-1} is introduced to emphasize that this exponential model explicitly takes the word classes as features. The goal is that, by grouping features of similar n-grams (having the similar predicted word w_n in the same class k_n or having the similar history words w_1^{n-1} in the same set k_1^{n-1}), the model size can be reduced, leading to improvements as claimed by the above heuristic.

The quality of word classes has an important influence on the performance of the resulting model. Similarly to the clustering problem in class-based approaches, finding the optimal word classes in the likelihood framework of Model Ms is impractical due to its high computational demand. In the original work, the algorithm of Brown et al. (1992) was used. Recently, a modified version of this clustering algorithm that is specifically tailored to Model M was developed in (Chen and Chu, 2010). Firstly, it modifies the objective function by replacing the conditional bigram probability in Equation (1.36) by a joint probability, then adding the same term but of the order 3:

$$\begin{aligned}\mathcal{L}^{\text{joint}}(D_{\text{train}}) = &\sum_{w_{n-1}w_n} c(w_{n-1}w_n) \log P(w_{n-1}w_n) \\ &+ \sum_{w_{n-2}w_{n-1}w_n} c(w_{n-2}w_{n-1}w_n) \log P(w_{n-2}w_{n-1}w_n) \end{aligned} \tag{1.62}$$

This term was shown to improve the quality of the word clustering used by Model M. Moreover, as Model M uses n-gram word contexts when predicting words and classes, n-gram features seem to be useful for class induction. Therefore, it is logical to use word n-gram probabilities and to back-off to class n-gram probabilities only for n-grams that do not occur in the training data. In this way, instead of using training set likelihood as a criterion, word classes are found by maximizing the test likelihood estimated using one type of smoothing techniques.

In large scale ASR tasks (Chen, 2009b; Chen et al., 2009), via lattice rescoring, model M has been shown to consistently achieve significant improvements over back-off language models. Essentially, model M differs from other exponential approaches in the way that it can take into consideration the similarity between words in both the context and prediction sides. This is done by using features induced not only from words but also from word classes. Similarly to exponential language models, their major weakness is the training complexity because of the enormous number of features which are not only defined over word n-grams but also over class n-grams.

In summary, all the sophisticated approaches to language modeling that have been described so far are based on statistics of discrete symbols, and have to deal with the *data sparsity* problem even in the presence of very large training corpora. This particular problem also referred to as *the curse of dimensionality* is used to describe the fact that with the higher dimensionality of the input space, one always needs much more data in order to make the learning reliable. In language modeling, the input space is often estimated through the vocabulary size (denoted by $|\mathcal{V}|$ or $\mathcal{V}$). For example, if $\mathcal{V}$ =10,000, even using the n-gram assumption with $n = 4$, the number of possible n-grams in theory is 10^{16} and hence the number of free parameters that needs to be trained is of the same order. There is never enough data to guarantee that all possible n-grams are seen at least once in this case. In practice, as a consequence of the Zipf's law (Zipf, 1932), the available corpora are usually limited to several billion (10^9) words with a lot of repeated n-grams. This illustrates why the data sparsity problem remains challenging even with the use of smoothing techniques, word classes . . . The number of free parameters is still too large. The information for less frequent n-grams is still difficult to capture, leading to an underestimation of their probabilities.

1.4 Continuous Space Language Models

As shown in the previous sections, approaches to language modeling related to considering words as discrete symbols all suffer from the data sparsity problem. One of the most successful alternatives to date which aims to use *distributed word representations* to remedy this issue was introduced in (Bengio, Ducharme, and Vincent, 2000), where distributionally similar words are represented as neighbors in a continuous space. This turns n-grams distributions into smooth functions of the word representations. More concretely, each word in the vocabulary is mapped into a real-valued vector and the conditional probability distributions are then expressed as a (parameterized) smooth function of these feature vectors.

The similarity between words in a continuous space, if successfully learned, can generalize from the observation of frequent n-grams to similar instances of less frequent (or even unseen) n-grams, thereby reducing the sparsity issues that plague conventional maximum likelihood estimation. The underlying idea is that if "similar" word sequences have similar representations; a small difference between their representations will only imply a small variance in the estimated probability thanks to the smoothing property of functions. Moreover, when one example is encountered in the training step, the model will try to learn not only information about this example but also about similar ones. For example, in the training data, we have seen this sentence: *"Alice is the only girl that Bob loves"*. It should help to predict the probability of the new sentence which we assume, doesn't occur in the training set: *"Carole is the only woman that Bob loves"* as long as the continuous representations for the couple of words (*"Alice"*, *"Carole"*) are close and likewise for (*"girl"*, *"woman"*). To achieve this goal, the choice of the mapping plays a crucial role, since it needs to successfully encode similarity relationships between words. Moreover, the smoothness assumption needs to be fulfilled by the distributed representations induced from the mapping as it serves as an input of the function estimation.

In (Emami, 2006), the author asserts that from the point of view of the curse of dimensionality, by mapping random variables from the original high-dimensional discrete space ($\mathcal{V}$) into a low dimensional but continuous one, *"the estimation problem is not as severely crippled by the curse of dimensionality associated with high-dimensional discrete random variables"*. In most of the case, even for large scale tasks, continuous space language models have fewer parameters than discrete ones. Their number of free parameters is typically in the order of $|M\mathcal{V}|$ (where M is the dimension of a continuous space that is always much smaller than $\mathcal{V}$), therefore being almost linear in the vocabulary size. For comparison, the number of discrete n-gram word-based language models is in theory in order of $\mathcal{V}^n$, in practice much smaller, in order of the number of n-gram types in the training data that is proportional to the number of running words in the training data.

The important advantage of the continuous space based approach is that the representations of words in a continuous space are learned automatically. In the original article (Bengio, Ducharme, and Vincent, 2000), the projection of words into a continuous space and the associated probability estimates were jointly carried out in a multi-layer neural network architecture. This idea was inspired from other words, such as (Hinton, 1986; Elman, 1990; Paccanaro and Hinton, 2001) where distributed representation for discrete symbols were used within a neural network framework. Applying neural networks in language modeling was also investigated in several articles such as (Nakamura et al., 1990; Miikkulainen and Dyer, 1991; Schmidhuber and Heil, 1996; Xu and Rudnicky, 2000). However, in (Bengio, Ducharme, and Vincent, 2000), it was the first time that the idea of using neural network language models was pushed to a large scale. We will call from now all models based on this approach as Neural Network Language Models (NNLMs for short). The formalism of neural networks allows us to express them in a unified framework, where, crucially, word representations are learned in conjunction with the other model parameters. The choice of the mapping is completely ignored since the word representations are learned automatically for a particular estimation task.

This approach has shown significant and consistent improvements when applied to automatic speech recognition (Schwenk, 2007; Emami and Mangu, 2007; Kuo et al., 2010) and machine translation tasks (Schwenk, Dchelotte, and Gauvain, 2006). Hence, continuous space language models have become increasingly used. These successes have revitalized the research on neuronal architectures for language models, and given rise to several new proposals (see, for instance, (Mnih and Hinton, 2007; Mnih and Hinton, 2008; Collobert and Weston, 2008; Mikolov et al., 2010)). A major difficulty with these approaches remains the complexity of training, which does not scale well to the massive corpora that are nowadays available. Practical solutions to this problem are discussed in (Schwenk, 2007), which introduces a number of optimizations and tricks to make training tractable. Even then, training a neuronal language model typically takes days.

The remaining of this section is organized as follows. In Section 1.4.1, five types of neural network language models are described. Then, the training algorithm for feed-forward models is presented in Section 1.4.2. The model complexity in terms of the number of parameters and the computational time is analyzed in Section 1.4.3. The way to introduce the score of NNLMs into practical tasks (ASR, SMT) is finally described in Section 1.4.4.

1.4.1 Current Approaches

Here, we are going to describe several variations of continuous space language models considered in our study and discuss the various issues associated with the training of such models, as well as their most common remedies. Let us first introduce some notations in Table 1.1.

1.4.1.1 Standard Feed-forward Models

In the following, we will consider words as indices in a finite dictionary of size $\mathcal{V}$; depending on the context, w will either refer to the word or to its index in the dictionary. A word w can also be represented by a 1-of-V coding vector $\mathbf{v}$ in a high-dimensional discrete space $\mathbb{R}^V$ in which all components are null except the w^{th} which is 1. In the standard approach of (Bengio, Ducharme, and Vincent, 2000), the feed-forward network which will be referred to as *standard NNLM*, takes as input the $n-1$ word history w_1^{n-1} and delivers an estimate of the probability $P(w_n|w_1^{n-1})$ as its output. It consists of three layers as represented in Figure 1.2.

The input layer builds a continuous representation of the history by mapping each word into its real-valued representation. This mapping is defined by $\mathbf{R}^T\mathbf{v}$, where $\mathbf{R} \in \mathbb{R}^{V\times M}$ is a *projection matrix* and M is the dimension of the continuous projection word space. The output of this layer is a vector $\mathbf{i}$ of $(n-1)M$ real numbers obtained by concatenating the representations of the context words.

$$\mathbf{i} = \{\mathbf{R}^T\mathbf{v_1}; \mathbf{R}^T\mathbf{v_2}; \ldots; \mathbf{R}^T\mathbf{v_{n-1}}\} \tag{1.63}$$

The projection matrix $\mathbf{R}$ is shared along all positions in the history vector and is learned automatically.

The hidden layer introduces a nonlinear transform, where the output layer activation values are defined by

$$\mathbf{h} = f\left(\mathbf{W^h}\mathbf{i} + \mathbf{b^h}\right), \tag{1.64}$$

notation	meaning
$\mathbf{a}, \mathbf{b}, \mathbf{c} \dots$	vectors
$a_i, b_i, c_i \dots$	the i^{th} component of the vector
$\mathbf{A}, \mathbf{B}, \mathbf{C} \dots$	matrices
$A_{i,j}, B_{i,j}, C_{i,j} \dots$	the cell (i, j) of the matrix
$A_{:,j}, B_{:,j}, C_{:,j} \dots$	the column j of the matrix
$A_{i,:}, B_{i,:}, C_{i,:} \dots$	the line i of the matrix
w	a word; its index in the dictionary
V	context vocabulary or general vocabulary; the vocabulary size
M	projection dimension
n	model order
$\mathbf{R} \in \mathbb{R}^{V \times M}$	projection space
$\mathbf{v} \in \mathbb{R}^{V}$	1-of-V coding vector for words in the vocabulary (all null except the w^{th} $(= 1)$
$\mathbf{i} \in \mathbb{R}^{(n-1)M}$	input layer
f	a nonlinear function (sigmoid or tangent hyperbolic)
H	hidden layer size
$\mathbf{W^h} \in \mathbb{R}^{H \times (n-1)M}$	weight matrix between the input layer and the hidden layer
$\mathbf{b^h} \in \mathbb{R}^{H}$	bias vector for the hidden layer
$\mathbf{h} \in \mathbb{R}^{H}$	hidden layer
V^o	output vocabulary; output vocabulary size
$\mathbf{W^o} \in \mathbb{R}^{V^o \times H}$	weight matrix between the hidden layer and the output layer
$\mathbf{b^o} \in \mathbb{R}^{V^o}$	bias vector at the output layer
$\mathbf{o} \in \mathbb{R}^{V^o}$	output layer (before *softmax* function)
$\mathbf{p} \in \{0, 1\}^{V^o}$	output layer (after *softmax* function)

Table 1.1: *Notations*

where $\mathbf{i}$ is the input vector, $\mathbf{W^h} \in \mathbb{R}^{H \times (n-1)M}$ and $\mathbf{b^h} \in \mathbb{R}^{H}$ are the parameters (weights and biases) of this layer. We use here f to denote a nonlinear function. For instance, it can be tangent hyperbolic, $\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$ or sigmoid function, $\text{sigm}(x) = \frac{1}{1 + \exp(-x)}$. The vector $\mathbf{h} \in \mathbb{R}^{H}$ can be considered as a more abstract representation of the context than $\mathbf{i}$.

The output layer consists of V nodes, each node is associated with one word in the vocabulary. Its activation values can be computed using the following equation:

$$\mathbf{o} = \mathbf{W^o}\mathbf{h} + \mathbf{b^o}, \tag{1.65}$$

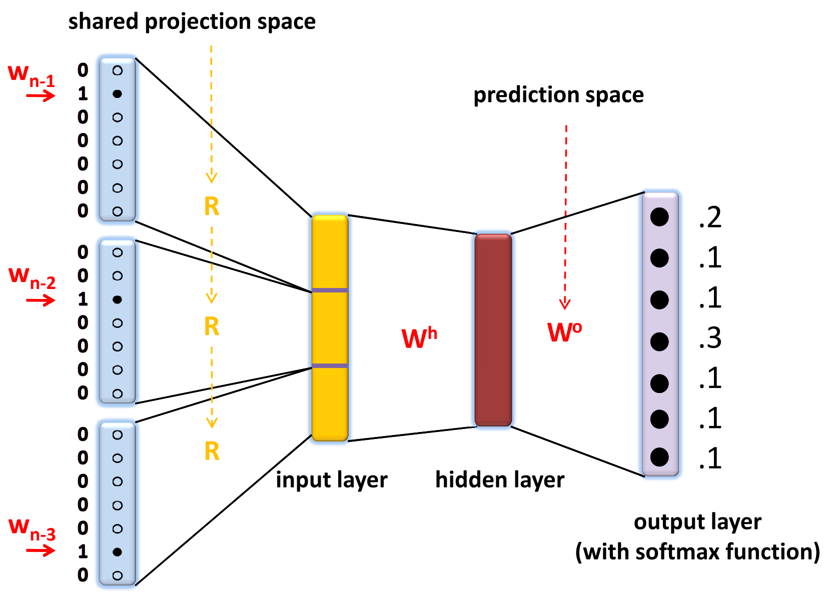


Figure 1.2: *Feed-forward neural network language model architecture.*

where $\mathbf{W^o} \in \mathbb{R}^{V^o \times H}$ and $\mathbf{b^o} \in \mathbb{R}^{V^o}$ are respectively the projection matrix and the bias term associated with this layer. Here, we use V^o rather than V to denote the output vocabulary because in general, the output and the input vocabularies can be different.

Then we can estimate the desired probability, thanks to the *softmax* function:

$$p_i = \frac{\exp(o_i)}{\sum_j \exp(o_j)} \tag{1.66}$$

The i^{th} component of $\mathbf{p}$ corresponds to the estimated probability of the i^{th} word of the vocabulary given the input history vector $P(w_n = i|w_1^{n-1})$.

For later use, we note here that in general, we use the term *softmax layer* to denote a layer that takes a hidden vector $\mathbf{h}$ as input to compute its output vector $\mathbf{p}$, a probability distribution.

The standard model has two hyper-parameters (the dimension of the projection space, M and the size of the hidden layer, H) that define the architecture of the neural network. It has a set of free parameters Θ that need to be learned from the data: the projection matrix $\mathbf{R}$, the weight matrix $\mathbf{W^h}$, the bias vector $\mathbf{b^h}$, the weight matrix $\mathbf{W^o}$ and the bias vector $\mathbf{b^o}$.

In this model, the projection matrices $\mathbf{R}$ and $\mathbf{W^o}$ play similar roles as they define maps between the vocabulary and the hidden representation. The fact that $\mathbf{R}$ assigns similar representations to history words w_1 and w_2 implies that these words can be exchanged with little impact on the resulting probability distribution. Likewise, the similarity of two lines in $\mathbf{W^o}$ is an indication that the corresponding words tend to have a similar behavior, i.e., tend to have a similar probability of occurrence in all contexts. In the remainder, we will therefore refer to **R** as the matrix representing the *input (context) space*, and to $\mathbf{W^o}$ as the matrix for the *output (prediction) space*.

1.4.1.2 Log-bilinear Models

The work reported in (Mnih and Hinton, 2007) describes another parameterization of the architecture introduced in the previous section. This parameterization is based on Factored Restricted Boltzmann Machine. According to (Mnih and Hinton, 2007), this model, termed the *log-bilinear* language model (LBL), achieves, for large vocabulary tasks, better generalization in terms of perplexity than the standard model, even if the reasons beyond this improvement remain unclear. In this section, we will describe this model and show how it relates to the standard model. The LBL model estimates the n-gram parameters by:

$$P(w_n|w_1^{n-1}) = \frac{\exp(-E(w_n;w_1^{n-1}))}{\sum_w \exp(-E(w;w_1^{n-1}))} \tag{1.67}$$

In this equation, E is an *energy function* defined as:

$$\begin{aligned} E(w_n;w_1^{n-1}) &= -\left(\sum_{j=1}^{n-1} \mathbf{v_j}^T \mathbf{R}\mathbf{W_j}^T\right)\mathbf{R}^T\mathbf{v_n} - \mathbf{b_r}^T\mathbf{R}^T\mathbf{v_n} - \mathbf{b_v}^T\mathbf{v_n} \\ &= -\mathbf{v_n}^T\mathbf{R}\left(\sum_{j=1}^{n-1}\mathbf{W_j}\mathbf{R}^T\mathbf{v_j} + \mathbf{b_r}\right) - \mathbf{v_n}^T\mathbf{b_v}, \end{aligned} \tag{1.68}$$

where **R** is the projection matrix introduced above, $\mathbf{v_j}$ are the 1-of-V coding vector for the j^{th} previous words and $\mathbf{v_n}$ is the coding vector for w_n; $\mathbf{W_j} \in \mathbb{R}^{M\times M}$ is a combination matrix and $\mathbf{b_r}$ and $\mathbf{b_v}$ denote bias vectors. All these parameters need to be learned during training.

Equation (1.68) can be rewritten using the notations introduced for the standard model. We then rename $\mathbf{b_r}$ and $\mathbf{b_v}$ respectively $\mathbf{b^h}$ and $\mathbf{b^o}$. We also denote **i** the concatenation of the $(n-1)$ vectors $\mathbf{R}^T\mathbf{v_j}$; likewise $\mathbf{W^h}$ denotes the $H \times (n-1)M$ matrix obtained by concatenating row-wise the $(n-1)$ matrices $\mathbf{W_j}$. With these new notations, the computing procedure reflected by Equations (1.67) and (1.68) can be reformulated as:

$$\mathbf{h} = \mathbf{W^h}\mathbf{i} + \mathbf{b^h}, \tag{1.69}$$

$$\mathbf{o} = \mathbf{R}\mathbf{h} + \mathbf{b^o}, \tag{1.70}$$

$$p_i = \frac{\exp(o_i)}{\sum_j \exp(o_j)} \tag{1.71}$$

This formulation highlights the similarity between the LBL model and the standard model. These two models differ only by the activation function of their hidden layer (linear for the LBL model and nonlinear for the standard model) and by their definition of the output space: for the LBL model, the input space and the output space are the same ($\mathbf{R} = \mathbf{W^o}$). H must therefore be equal to M. Moreover, there is only one vocabulary $V = V^o$. On the contrary, in the standard model, the output space is defined independently from the input space. This restriction drastically reduces the number of free parameters of the LBL model.

It is finally noteworthy to outline the similarity of this model with standard maximum entropy language models that are described in Section 1.3.7. Let $\mathbf{x}$ denote the binary vector formed by stacking the $(n-1)$ 1-of-V encodings of the history words; then the conditional probability distributions estimated in the model are proportional to $\exp(F(\mathbf{x}))$, where F is an affine transform of $\mathbf{x}$. The main difference with maximum entropy language models are thus the restricted form of the feature functions, which only test one history word, and the particular representation of F, which is defined as:

$$F(\mathbf{x}) = \mathbf{R}\mathbf{W^h}\mathbf{R'}^T\mathbf{v} + \mathbf{R}\mathbf{b^h} + \mathbf{b^o}, \tag{1.72}$$

where, as before, $\mathbf{R'}$ is formed by concatenating $(n-1)$ copies of the projection matrix $\mathbf{R}$.

1.4.1.3 Hierarchical Log-bilinear Models

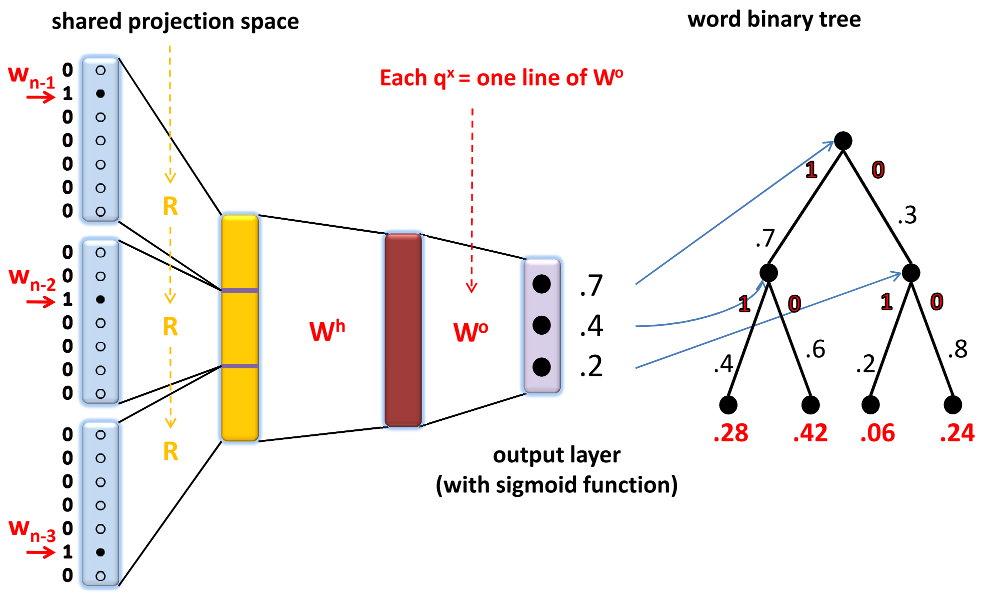


Figure 1.3: *Hierarchical log-bilinear model architecture.*

According to (Morin and Bengio, 2005), it is possible to replace the output layer of NNLMs by a hierarchical structure obtained from the expert knowledge (WordNet). However, this approach degrade the performance. In (Mnih and

Hinton, 2008), the authors proposed to adopt this strategy to LBL models. The resulting model as depicted in Figure 1.3 is called *hierarchical log-bilinear model* (HLBL). The output layer is structured by grouping words using a binary tree. It is constructed automatically from the training data.

First, each predicted word w_n is represented as a leaf in the tree. We denote the sequence of nodes from the root to its leaf by $x_0^U = x_0 x_1 \ldots x_U$ where x_0 is the root and U is the depth of the tree. The path from the root to its leaf is associated with an array of binary values $\zeta_1^U = \zeta_1 \zeta_2 \ldots \zeta_U$. We use $x_u = \zeta_{u+1}$ to represent the events that at node x_u, we turn left (if $\zeta_{i+1} = 1$) or we go to the right (otherwise).

For each node x in the tree, we can compute the probability of going to the left using one output node of a neural network. First, we compute the hidden layer of the neural network **h** from the indices of history words in the same way as with the standard NNLM and the LBL. Then, we use the sigmoid function to compute the following probability:

$$P(x = 1 | w_1^{n-1}) = \text{sigm}(\mathbf{q^x}^T \mathbf{h} + b^x), \tag{1.73}$$

where $\mathbf{q^x}$ and b^x are in fact the weight vector and the bias value associated with node x. This equation assumes that each node can be predicted independently of the others.

Now, the probability of word w_n given its history w_1^{n-1} is first decomposed using its binary code to be:

$$P(w_n | w_1^{n-1}) = P(\zeta_1 | w_1^{n-1}) \prod_{u=2}^{U} P(\zeta_u | w_1^{n-1}, \zeta_1^{u-1}) \tag{1.74}$$

As ζ_1^{u-1} is used to encode a path in the tree, from the root to node x_{u-1}, this node x_{u-1} can be used to represent ζ_1^{u-1}. Each conditional probability in the previous equation is then computed as follows[3]:

$$\begin{aligned} P(\zeta_u | w_1^{n-1}, \zeta_1^{u-1}) &= P(\zeta_u | w_1^{n-1}, x_{u-1}) \\ &= P(x_{u-1} = \zeta_u | w_1^{n-1}) \\ &= \text{sigm}(\mathbf{q^{x_{u-1}}}^T \mathbf{h} + b^{x_{u-1}}) \end{aligned} \tag{1.75}$$

With this type of model, we use a tree structured representation of words, therefore, a softmax function is not required and the computational time is significantly reduced: the multiclass prediction of the softmax is replaced by a bunch of binary predictors. Compared to LBL models, the output word space no longer exists. This work can be considered as an extension of the idea of factoring the probability for exponential models to multiple levels (Section 1.3.7). Using a binary tree to factorize the computation, for each example, the computational time is linear to $\log_2(\mathcal{V}^o)$ in place of $\mathcal{V}^o$. The

[3]To deal with the first term, node x_0, the root of the tree, is used in the following equation.

resulting speed-up in training reported in (Mnih and Hinton, 2008) is thus significant: a factor greater than 250 but at the price of a 13% perplexity increase. The reason of this loss may be due to the recursive binary structure. If one word is poorly clustered, this error affects all the internal nodes (or clusters) which lead to this word. This is typically the case for rare words that represent most of the vocabulary. Therefore an error in one word may have a significant impact on the whole system.

1.4.1.4 Recurrent Models

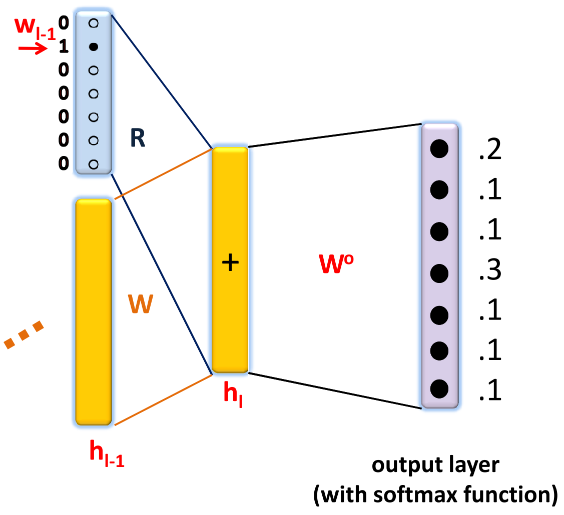


Figure 1.4: *Recurrent neural network language model architecture.*

Recurrent neural network language models (*RNNLMs*) are based on a more complex architecture, an Elman network (Elman, 1990), designed to recursively handle an arbitrary number of context words. Described for the first time in (Mikolov et al., 2010), they were shown experimentally to outperform both standard back-off LMs and feed-forward NNLMs in terms of perplexity on a small task. In (Mikolov et al., 2011a), the experimental results for a small ASR task showed that RNNLMs significantly outperform a large number of advanced language modeling techniques, including class-based, cache, exponential LMs, feed forward NNLMs . . . both in terms of perplexity and WER.

The key aspect of this architecture as reflected in Figure 1.4 is that the hidden layer $\mathbf{h_l}$ for predicting the l^{th} word w_l in a text is computed from both a numeric representation of the previous word w_{l-1} ($\mathbf{R}^T\mathbf{v_{l-1}}$ as in the standard model) and from the hidden layer for the previous word ($\mathbf{h_{l-1}}$) using the following recursive equation:

$$\mathbf{h_l} = f(\mathbf{W}\mathbf{h_{l-1}} + \mathbf{R}^T\mathbf{v_{l-1}}) \quad (1.76)$$

The hidden layer thus acts as a representation of the context history that iteratively accumulates an unbounded number of previous words representations.

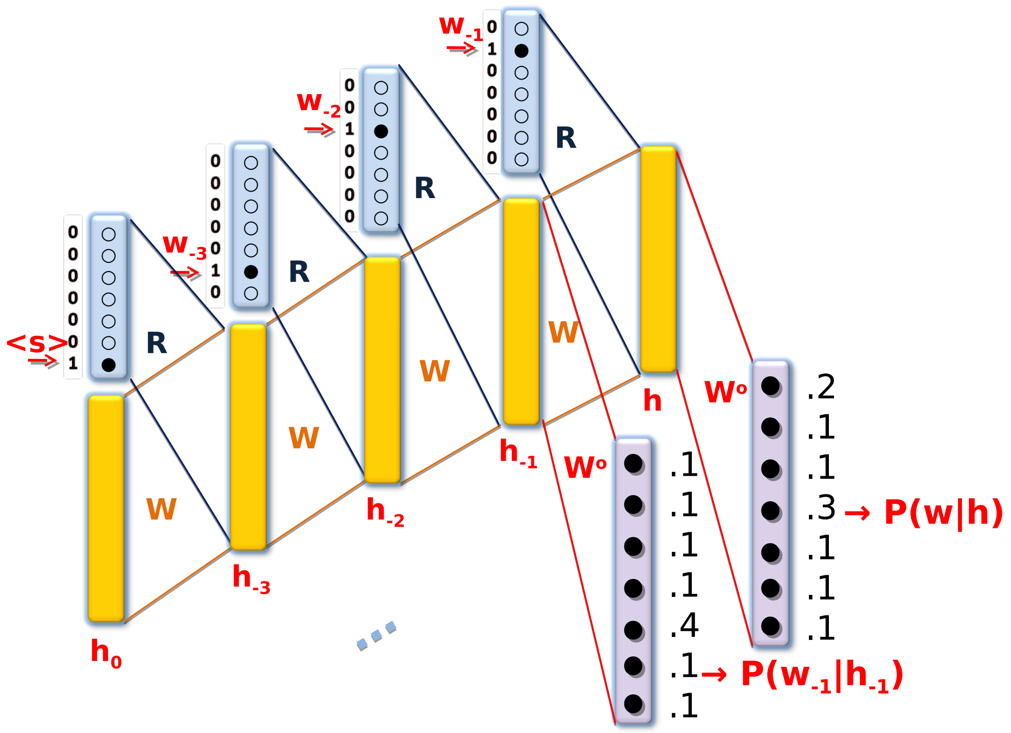


Figure 1.5: *Unfolded recurrent neural network language model architecture.*

To better understand the behavior of RNNLMs, we unfold their architecture as shown in Figure 1.5. Starting with <s>, the first word of a document, we use the initial value $\mathbf{h_0}$ of the hidden layer and the representation of <s> to calculate $\mathbf{h_1}$. To predict w_l, we repeat this operation until we reach $\mathbf{h_l}$, using this value to compute the values of the output layer as in feed forward NNLMs. As usual, $\mathbf{R}^T\mathbf{v_i}$ associates each context word $\mathbf{v_i}$ to one feature vector (the corresponding row in $\mathbf{R}$). This vector now plays the role of a bias vector at subsequent hidden layers. The first part (before the output layer) is thus structured in a series of layers. In case of predicting w_l, the relation between the hidden layer $\mathbf{h_l}$ and the first previous word w_{l-1} is at level 1 (with a direct connection), the second previous word w_{l-2} is at level 2 (through the previous hidden layer $\mathbf{h_{l-1}}$) and so on.

For inference, we compute the probabilities of a test text exactly in this fashion, proceeding recursively from the beginning of the text. For each word, we need to accomplish the operation of Equation (1.76) to compute the hidden layer.

Recurrent models are trained in the same way as feed-forward NNLMs, with the stochastic back propagation algorithm (described in detail in Section 1.4.2). In (Mikolov et al., 2010), at each training step, by restricting the error to be propagated through $\mathbf{h_{n-1}}$, the links between the current hidden layer and other context words and histories further than $l-1$ positions are ignored. Therefore, it is possible to update $\mathbf{Rv_{n-1}}$, the representation of w_n, and $\mathbf{W}$, the weight matrix between hidden layers, only in order to capture the relation between two consecutive positions. By using *Back-Propagation Through Time* (BPTT) (Rumelhart, Hinton, and Williams, 1986), in (Mikolov et al., 2011b), the authors showed that it is better to propagate the error back to a deeper level than 1, at least to $\mathbf{h_{l-4}}$. It means that the recurrent architecture is in fact not as compact as reflected by Figure 1.4 because the parameters $\mathbf{W}$ and $\mathbf{R}$ are actually learned so as to capture the relation between multiple word positions as is better illustrated in Figure 1.5. Therefore, RNNLMs differ from feed-forward NNLMs crucially in the way they modify the relation between the hidden layer and word representations, somehow deeper. That is the reason why in Chapter 5, we propose to use an approximate version of RNNLMs to speed up the training procedure.

Significant improvements in ASR using these models were reported in (Mikolov et al., 2011c; Mikolov et al., 2011b). However, similar to feed-forward networks, training and inference procedures are very expensive (the complexity of NNLMs are analyzed below, in Section 1.4.3). To speed up, the authors propose a hierarchical architecture similar to the exponential and HLBL models, based however on a simple unigram clustering. For large scale tasks ($\approx 400M$ training words), advanced training strategies were investigated in (Mikolov et al., 2011b). The data was divided into paragraphs, filtered and then sorted: the most in-domain data was thus placed at the end of each epoch. Moreover, the hidden layer size was decreased by simulating a maximum entropy model using a hash function on n-gram features. This part represents direct connections between the discrete 1-of-V vectors representations of context words and the output layer. By sharing the prediction task, the work of the hidden layer is made simpler, and can thus be handled with a smaller number of hidden units. Note that, this approach reintroduces into the model discrete features which are somehow a main weakness of conventional back-off LMs as compared to NNLMs. In fact, this strategy can be viewed as an effort to directly combine the two approaches (back-off and neural network LMs), instead of using a more direct approach through linear interpolation. Training simultaneously two different models is computationally very demanding for large vocabularies, even with the help of hashing techniques.

1.4.1.5 Ranking Language Models

In (Collobert and Weston, 2008; Collobert et al., 2011), the authors proposed to use a unified neural network architecture to do several natural language processing tasks including a model implementing a different approach to language modeling, namely a *ranking language model*. Instead of estimating distributions of n-grams, inspired by pairwise-ranking approach, the goal of this new type of language model is to provide scores that describe the acceptability of a piece of text. The other tasks are:

- Part of Speech (POS): Label each word with a unique tag that indicates its syntactic role, e.g., plural noun, adverb ...
- Chunking (shallow parsing): Label segments of a sentence with syntactic constituents: noun or verb phrases (NP or VP) ... tags are: begin-chunk (B-NP), inside-chunk tag (I-NP) ...
- Named Entity Recognition (NER): Label atomic elements in the sentence into "semantic" categories: "PERSON" or "LOCATION" ...
- Semantic Role Labeling (SRL): Give a semantic role to a syntactic constituent of a sentence, e.g., "[John]ARG0 [ate]REL [the apple]ARG1"

The main goal of this proposition is to learn internal word representations from mostly unlabeled training data through ranking language models, which can then deliver useful information for other tasks. Most importantly, these representations are automatically learned thus avoiding task-specific engineering, disregarding a lot of prior knowledge. The same algorithm, i.e., stochastic gradient descent with back propagation was carried out following ranking criterion for the particular language model.

Note that, all tasks except language model use annotated training data. Existing approaches are often based on the use of task-specific *ad-hoc, hand-crafted, engineering* features derived from output of preexisting systems which the authors of (Collobert and Weston, 2008) claim to be complex and slow. The second goal is therefore, to try to use as few features of this type as possible, notably, to efficiently take into account learned word representations. To achieve this goal, similar neural network architectures are used for all tasks. There are two approaches: *window* which means that models can take as input features from several neighbors of the current word; *sentence* which means that the input features are from the entire current sentence.

Window based neural networks, as depicted in the left side of Figure 1.6, consist of three main parts. At the beginning, features are created by projecting discrete variables such as surface word form, word-stemming, upper or lower case, prefix, suffix ... into their own continuous space. As with NNLMs, all the representations for each value of each discrete feature are learned simultaneously with the other parameters of the models so as to maximize their objective function. A hidden layer with nonlinear activation (*HardTanh* which is similar to *tanh*) is then added. At the output, there is a layer whose size is equal to number of output tags except for the ranking language model for which is only one node (as seen later). Finally, a normalization softmax function is applied.

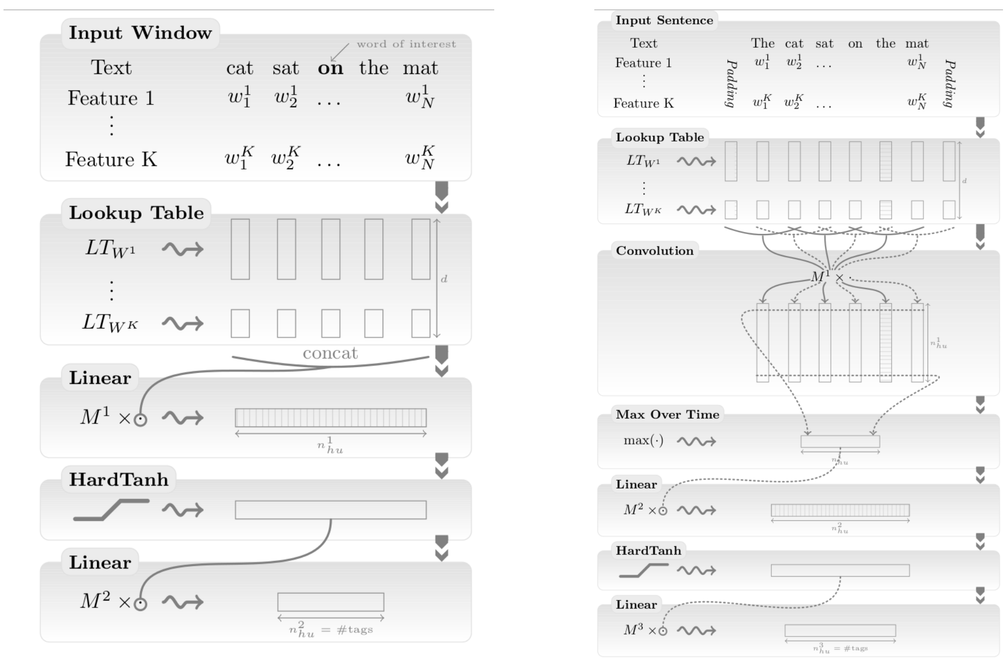


Figure 1.6: *Window (left) and sentence (right) approaches. This figure is borrowed from (Collobert et al., 2011).*

The window approach fails to do SRL, the most complex task. Therefore, the authors need to use sentence based neural networks (the right part of Figure 1.6). They are more complicated as they integrate features are from the words of the whole sentence. Now, the word needed to be tagged can no longer implicitly be recognized as the middle word of the window. A solution is to provide for each word in the sentence a special feature that encodes their relative position with respect to the word of interest. After projecting discrete features into their spaces, there is a *convolution layer* dealing with several sliding windows. Letting z be the size of features after the convolution layer for each window, x be the number of windows, we can use a matrix $\mathbf{X} \in \mathbb{R}^{z \times x}$ to represent the output of the convolution layer. It is worth noticing that x depends on arbitrary sentence lengths. Therefore, a *Max Over Time operation* is applied for each element i:

$$y_i^{\max} = \max\left[\mathbf{X}_{i,:}\right], \tag{1.77}$$

$$y_i^{\text{argmax}} = \text{argmax}\left[\mathbf{X}_{i,:}\right] \tag{1.78}$$

Note that, the output vector $\mathbf{y}^{\mathbf{max}} \in \mathbb{R}^z$ with a fixed length is fit to apply subsequent standard layers. By counting the number of times the index of each window occurs in $\mathbf{y}^{\mathbf{argmax}}$, we can rank windows in terms of their importance to the prediction. Thus, selecting the most useful features is another capacity of the max layer. As in the window approach, some other layers are added to finally obtain a score vectors of tags.

The global systems are processed similarly to Hidden Markov Models (HMMs) where the emission scores are estimated with a sentence based neural network. They also have initial scores and transition scores (to be learned separately) in order to capture the relation between two consecutive tags. For inference, Viterbi algorithm is typically used.

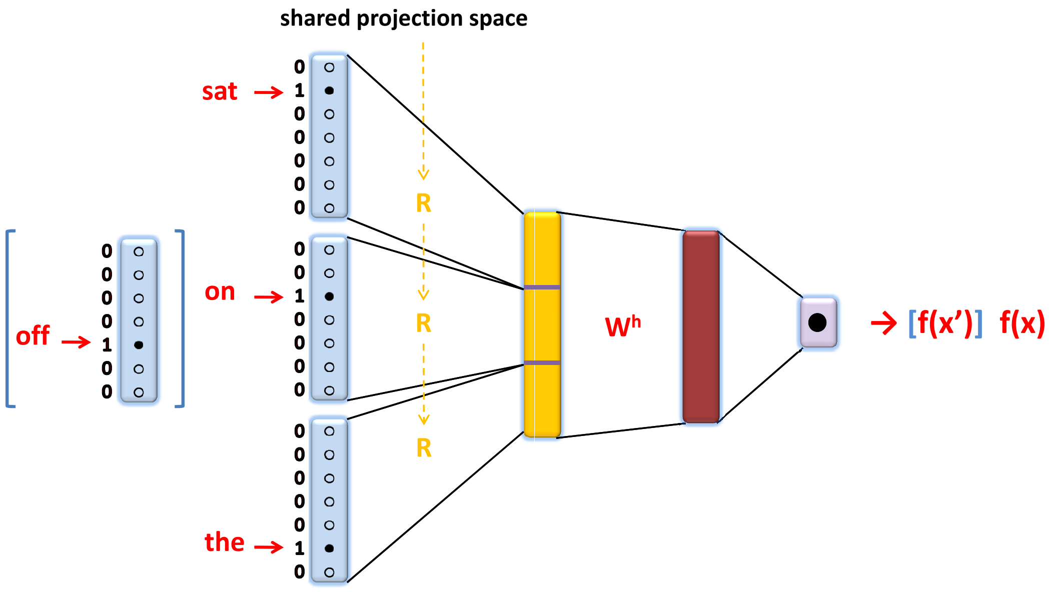


Figure 1.7: *Ranking neural network language model architecture.*

We consider now in detail ranking language models (represented on Figure 1.7) that are directly related to our work. Based on pair-wise ranking approach, they use the window-based approach in order to score the acceptability of a piece of text by positive and negative examples. Positive examples are obtained from the training corpus, whereas negative examples are artificially created by replacing the central word by any another word, drawn from a uniform distribution. In the figure, “sat on the” is the positive example, “sat off the” is the negative example obtained by substituting “on” by “off”.

To avoid the expensive softmax function, they use a simple objective function:

$$\max\left[0, 1 - \text{Scr}(x) + \text{Scr}(x')\right], \tag{1.79}$$

where Scr is the output of neural network, x, x' are respectively positive and negative examples which only differ in the middle word. The main goal is to separate positive and negative examples by giving a larger score for positive ones with a limit of difference, which is 1. Each training example is a couple of positive example and its associated negative instance.

The experimental results suggest that *word embeddings* (word representations) when learned with ranking language models have some meaning. Words that are similar are usually grouped together. Conversely, models for other tasks when trained separately cannot provide any comprehensible representations. Using the word embeddings of the ranking language models as initialization values for word embeddings of other tasks significantly increased the system performance. In all tasks, the final neural network models achieve equivalent or even better results in comparison with other state-of-the-art systems.

1.4.2 Training algorithm

Training the neural network language models introduced above can be achieved by maximizing some objection functions such as the log-likelihood $\mathcal{L}$ of the parameters Θ in the case of NNLMs, RNNLMs and the pair-wise ranking score in the case of ranking models. This optimization is usually performed using stochastic back-propagation as in (Bengio, Ducharme, and Vincent, 2000). Learning starts with a random initialization of the parameters under a uniform distribution[4] and converges to a local maximum of the log-likelihood function. Moreover, to prevent overfitting, an early stopping strategy is usually adopted: the likelihood of a validation set is computed after each epoch, and the training procedure is halted if it stops increasing.

To make things clearer, we briefly present here the typical approach for training a feed-forward neural network language model using the stochastic gradient descent algorithm (Algorithm 1). It is worth noticing that the training for the other types of neural network models is also straightforward and obeys the same principles. In order to have more details about our implementation, the reader is invited to go to Appendix D.

1.4.2.1 An overview of training

The objective function considered here is as follows:

$$\mathcal{L}(\Theta) = \sum_{l} \log P(w_l | w_{l-n+1}^{l-1}) - \mu \times \mathcal{R}(\mathbf{W^o}, \mathbf{W^h}), \tag{1.80}$$

where $\mathcal{R}(\mathbf{W^o}, \mathbf{W^h})$ is a regularization term, typically the L2-norm of the weights of the hidden and output layers (the summation over square of each element of $\mathbf{W^o}$ and $\mathbf{W^h}$) and $\mu \geq 0$ is the *weight decay factor*. Regularization is used to avoid overfitting. The weight decay factor is only used when updating weights ($\mathbf{W^o}$ and $\mathbf{W^h}$).

[4]Originally, the biases of the output layer are initialized with unigram distribution of words estimated from the training data. The results of our experiments show that it is not so important.

Algorithm 1 Training procedure for feed-forward NNLMs

```
for Each epoch do
  Randomizing the order of $n$-gram examples
  for Each $n$-gram example do
    Updating the parameters
  end for
  if Validation perplexity increases then
    Stop
  end if
end for
```

1.4.2.2 The update step

Each parameter updating step consists of two passes: the *forward pass* and the *backward pass*. During the forward pass, the inference from the input layer to the output layer is carried out. At the end of this pass, the first term in the objective function for an n-gram example w_1^n is the output estimate of model and hence, computed using *softmax* function:

$$\begin{aligned} \log P(w_n|w_1^{n-1}) &= \log p_w & (1.81)\\ &= \log\left(\frac{\exp(o_w)}{\sum_i \exp(o_i)}\right) & (1.82)\\ &= o_w - \log\left(\sum_i o_i\right), & (1.83)\end{aligned}$$

where w is the index of w_n in $\mathcal{V}$.

To construct the backward pass, we denote the derivative of $\log P(w_n|w_1^{n-1})$ with respect to a variable $\lambda \in \Theta$ by Δ:

$$\Delta\lambda = \frac{\partial \log P(w_n|w_1^{n-1})}{\partial\lambda} \tag{1.84}$$

All derivatives are computed following the stochastic back-propagation algorithm, i.e., from the output layer to the input layer.

At the output layer: We have:

$$\Delta o_i = \begin{cases} 1 - p_i & \text{if } i = w \\ -p_i & \text{otherwise} \end{cases} \tag{1.85}$$

At the hidden layer: From the inference equation for the output layer:

$$\mathbf{o} = \mathbf{W^o h} + \mathbf{b^o} \tag{1.86}$$

This implies that:

$$\begin{aligned} \Delta\mathbf{b^o} &= \Delta\mathbf{o}, && (1.87)\\ \Delta\mathbf{W^o} &= \Delta\mathbf{o}\mathbf{h}^T, && (1.88)\\ \Delta\mathbf{h} &= \mathbf{W^o}^T\Delta\mathbf{o} && (1.89)\end{aligned}$$

At the input layer: The inference equation for the hidden layer:

$$\mathbf{h} = f(\mathbf{W^h}\mathbf{i} + \mathbf{b^h}) \qquad (1.90)$$

implies that:

$$\begin{aligned} \Delta[\mathbf{W^h}\mathbf{i} + \mathbf{b^h}] &= f'(\mathbf{h})\Delta\mathbf{h}, && (1.91)\\ \Delta\mathbf{b^h} &= \Delta(\mathbf{W^h}\mathbf{i} + \mathbf{b^h}), && (1.92)\\ \Delta\mathbf{W^h} &= \Delta(\mathbf{W^h}\mathbf{i} + \mathbf{b^h})\mathbf{i}^T, && (1.93)\\ \Delta\mathbf{i} &= \mathbf{W^h}^T\Delta(\mathbf{W^h}\mathbf{i} + \mathbf{b^h}), && (1.94)\end{aligned}$$

where f' is the derivative function of f:

$$f'(x) = \begin{cases} 1 - \tanh(x)^2 & \text{if } f = \tanh \\ \mathrm{sigm}(x) - \mathrm{sigm}(x)^2 & \text{if } f = \mathrm{sigm} \end{cases} \qquad (1.95)$$

After computing all derivatives, each parameter λ is then updated using a *learning rate parameter* $\xi > 0$:

$$\lambda = \begin{cases} \lambda + \xi(\Delta\lambda - \mu'\lambda) & \text{if } \lambda \in \mathbf{W^o} \text{ or } \mathbf{W^h} \\ \lambda + \xi\Delta\lambda & \text{otherwise} \end{cases} \qquad (1.96)$$

Note that $\mu' = 2\mu \geq 0$. In experimental setups, the value of the weight decay factor is thought to be the value of μ'.

While the value of μ' is fixed in the training step, the value of ξ is in fact a function of the number of examples that have been seen. There are two usual ways to update the learning rate. The first one is:

$$\xi = \frac{\xi_0}{1 + \nu N_e}, \qquad (1.97)$$

where N_e is the number of already seen examples, $\xi_0 > 0$ is the initial value of ξ which, in most setup descriptions, is usually referred to the learning rate, $\nu > 0$ is the *learning rate decay*. The second way to update ξ is to use validation data. At the first time, the learning rate is fixed to ξ_0. After each epoch, if the perplexity of validation data is decreased, we keep it the same rate, otherwise, we start to divide it by 2 for each remaining epoch.

In practice, the choice of the learning rate and the weight decay factor needs to be optimized for each task. It depends not only on the size of the data but also on the model structure. Following the first approach, we need to choose three hyper-parameters (ξ_0: learning rate, ν: learning rate decay and μ': weight decay factor) while with the second one, there are only two (ξ_0: learning rate and μ': weight decay factor)[5]. Within the language model framework, the final results obtained using these two methods are very similar. For more details, the readers are encouraged to go to Appendix D.

1.4.3 Model complexity

In comparison to n-gram back-off language models, the number of parameters for NNLMs is in general smaller. Conversely, to compute the probability for each event, where back-off models need only some table look-up operations, NNLMs need to perform an expensive sequence of several mathematical vector-matrix or matrix-matrix operations. Moreover, the difference is even more sensible important during the learning phase: For a back-off model, the training algorithm requires to go through the training data just once to count n-grams, whereas for an NNLM, several passes through the training data are needed; at each epoch, doing a forward (inference) and a backward pass for each example. The main problem with NNLMs, as compared to other approaches is therefore their computational complexity. We will present here a complexity analysis for standard NNLMs without forgetting to say that all resulting equations and conclusions can be easily generalized for other types of NNLMs without any problem.

1.4.3.1 Number of parameters

There are three main parts:

- The input layer: its number of parameters is related to the dimension of the projection space: $M \times V$.
- The hidden layer: it consists of two parts: $(n-1) \times M \times H$ for weights $\mathbf{W^h}$, H for biases $\mathbf{b^h}$.
- The output layer: there are two parts: $H \times V^o$ for the weight matrix $\mathbf{W^o}$, V^o for the bias $\mathbf{b^o}$.

It results in a total number of free parameters for NNLMs:

$$M \times V + ((n-1) \times M + 1) \times H + (H+1) \times V^o \tag{1.98}$$

It should be noted that the total number of parameters grows linearly with both the number of words in the vocabulary and with the context length.

[5] From now, when the second approach is followed, the learning rate decay is marked "not used".

1.4.3.2 Computational issues

According to (Schwenk, 2007), for standard feed-forward NNLMs, the number of floating point operations needed to predict the label of a single example is:

$$((n-1) \times M + 1) \times H + (H+1) \times V^o \tag{1.99}$$

A number of operations required for processing one instance during training (backward and forward passes) is larger but of the same order. The first term of the sum corresponds to the computation of the hidden layer and the second one to the computation of the output layer. The projection of the context words amounts to selecting one row of the projection matrix **R**, as the words are represented with a 1-of-V coding vector. We can therefore assume that the computation complexity of the first layer is performed in constant time.

Most of the computation time is thus spent in the output layer, which implies that the computing time grows linearly with the output vocabulary size. Training these models for large scale tasks is therefore challenging, and a number of simplifications and tricks have been introduced to make training and inference tractable (Schwenk and Gauvain, 2002; Schwenk, 2007).

Shortlist The most important computational effort of NNLMs is at the output layer where most of the calculation is carried out. A simple method to reduce the complexity is thus to reduce the size of the output vocabulary (Schwenk, 2007): rather than estimating the probability $P(w_n|w_1^{n-1})$ for all words in the vocabulary, we only estimate it for the most frequent words of the training set (in the so-called *shortlist*, a subset of the vocabulary). In this case, two vocabularies are considered, corresponding respectively to the *input vocabulary* $\mathcal{V}$ used to define the history; and to the *output vocabulary* $\mathcal{V}^o$. However, this method fails to deliver any probability estimate for words outside of the output vocabulary, meaning that a fall-back strategy needs to be defined for those words. In practice, *shortlist based NNLMs* are combined with a conventional n-gram model as described in (Schwenk, 2007). The output estimation must be normalized with a standard back-off LM in order to make predictions for words which are not in the shortlist. The probability normalization formula with a shortlist model is:

$$P(w|h) = \begin{cases} \hat{P}_N(w|h) \times \gamma(h) & \text{if } w \in \text{shortlist} \\ \hat{P}_B(w|h) & \text{otherwise} \end{cases}, \tag{1.100}$$

where w is the word to be predicted and h its $n-1$ word history, $\hat{P}_N$ is the probability of an in-shortlist word calculated with the shortlist NNLM, $\hat{P}_B$ is the probability assigned by the standard back-off n-gram LM. The scaling factor $\gamma(h)$ depends on the history h and is defined as:

$$\gamma(h) = \sum_{w \,\in\, \text{shortlist}} \hat{P}_B(w|h) \tag{1.101}$$

As both $\hat{P}_N$ and $\hat{P}_B$ are normalized to sum to one, the "combined" conditional distribution P is also normalized.

Note that, in (Park et al., 2010), the authors proposed another normalization way based on the introduction of an additional output layer node to estimate the probability mass of out-of-shortlist words.

Data resampling The training time is linear in the number of examples in the training data as shown in the main procedure (Algorithm 1). Training NNLMs on very large corpora, which, nowadays, comprise billions of word tokens, cannot be performed exhaustively and requires to adopt *resampling* strategies, whereby, at each epoch, the system is trained with only a small random subset of the training data. This approach enables to effectively estimate neural language models on very large corpora; it has also been observed empirically that resampling the training data can increase the generalization performance (Schwenk, 2007) since there may be an added noise to the learning procedure as a result of the difference of training examples for each epoch. It should be noted that as large corpora are always composed of several parts that differ in genre, field, source ... their resampling rates needs to be properly distinguished according to their relevance to a task. For example, within an ASR framework, the resampling rate should be chosen to be much larger for text transcripts than for texts which are crawled from the web. This choice has a great impact on the final results.

Bunch mode Additional speed-ups can be obtained by propagating several examples at once through the network (Schwenk and Gauvain, 2004). This "bunch mode" as it was originally called or "mini-batch" allows using matrix-matrix operations rather than vector-matrix operations to reduce both inference and training time. Matrix operations are thought to be more easily accelerated on advanced CPU architectures, such as with multi-threading using optimized mathematical library. If we use bs, the "block size" to represent the number of examples used for each update, significant speed-up improvement up to ten times can be observed with bs $= 128$, though this modification doesn't yield any reduction of the number of floating point operations. Taking the computation between the input layer and the hidden layer as an example, this method turns the matrix-vector operation of Equation (1.64) into matrix-matrix operations as follows:

$$\mathbf{H} = f\left(\mathbf{W^h}\mathbf{I} + \mathbf{B^h}\right), \tag{1.102}$$

where each column of $\mathbf{I} \in \mathbb{R}^{(n-1)M\times \text{bs}}$ is an input vector $\mathbf{i}$ of one example, $\mathbf{B^h} \in \mathbb{R}^{H\times \text{bs}}$ is created by duplicating the bias vector $\mathbf{B^h}$ for each column of the matrix. Each column of the resulting matrix, $\mathbf{H} \in \mathbb{R}^{H\times \text{bs}}$, is a hidden vector $\mathbf{h}$ of the associated example.

Recently, using optimal library for graphical cards has been shown to make NNLM systems even faster (Schwenk, Rousseau, and Attik, 2012).

Context grouping During inference, as probabilities of n-grams having the same context can be calculated at once, it is better to first create a list of all n-grams, grouping n-grams with the same history, forwarding through a neural network only once for each history.

1.4.4 NNLMs in action

The integration of NNLMs for large-scale tasks (both ASR and SMT) is far from easy, given the computational cost of computing n-gram probabilities, a task that is performed repeatedly during the search of the best hypothesis. One solution is to resort to a two-pass approach: the first pass uses a conventional back-off n-gram model to produce a set of most likely solutions; in the second pass, the NNLMs probabilities are in turn computed and then incorporated into a system and, as a result, the hypotheses are accordingly reordered.

There are two typical ways of combining several language models. The usual way is through linear interpolation where the optimal weights of language models are obtained by minimizing the perplexity of a validation data. The original scores of an original language model are replaced by the new interpolated scores. It is reasonable to update the interpolation coefficients between models to better capture the interaction between them. For ASR, as we have few models (usually two: a language model and an acoustic model), their weights are unchanged. On the contrary, within the SMT framework, the weights of log-linear models are re-optimized as the other feature weights with appropriate algorithms, such as Minimum Error Rate Training (MERT) (Och, 2003). For SMT, there exists an alternative procedure, which is to consider the output of NNLMs as a new type of model, adding them as new scores and re-optimizing the weights of log-linear models to rerank the original hypotheses. The interaction between different types of language models is directly optimized with respect to the application metrics.

A set of the most likely solutions can in fact take two forms: m-best list and word lattice. The first one is a list of m best output hypotheses with respect to a baseline system. The second representation is a graph where each arc corresponds to a subpart of a hypothesis and is weights by is partial score. When rescoring the lattice with a language model of a higher order, we need to add nodes and arcs to guarantee that all arcs starting from the same node have the same context information. This technique is called “lattice expansion” that can augment drastically the size of the lattice especially, when the new n-gram order is much larger than the original one. To fit the lattice into memory, some pruning techniques often need to be applied. They sometimes hurt the performance of the system significantly as some potential good hypotheses according to the new set of features have been already discarded. Lattice is a more compact representation as it contains a larger number of hypotheses, hence lattice rescoring tends to give a better improvement than m-best rescoring. When Maximum A Posteriori (MAP) decoding is used with m large enough, there are no significant difference between the two paradigms. Still, in an

ASR framework where consensus decoding is always used, the difference is non-negligible. Based on these considerations, we will mostly use lattice in ASR experiments and on the other hand, prefer *m*-best rescoring in SMT tasks.

1.5 Summary

In Table 1.2, we present a brief summary of all state-of-the-art approaches for language modeling that have been presented so far. The information displayed in this table is from a practical point of view. The second column **feature** is used to show which features are taken into account by each model. While some features are induced directly from the text data (words, prefixes, documents, word triggers ...), others are obtained from external tools (POS Tagger for tags ...) or being learned (exchange algorithm for classes, EM for hidden topics in topic based models ...). The third column **prob interp** reports whether that models have a probabilistic interpretation or are only empirically motivated. The use of word order is represented in the fourth column **word order** where "yes (flexible)" means that it is considered in the implicit or more general fashion. The next column **word similarity** is used to show that models can take into account the similarity between words or not. While "hard" means a hard clustering (words are assigned for one class), "soft" is used to represent a soft clustering, meaning that the similarity between words is defined and measured without class definition. The column **context** shows the practical limit of the context size that models can consider (reported in most articles). Note that, "whole sentence" ("whole document") means that the context is all words from the first word of the current sentence (document) to the previous word. Recurrent neural networks can in theory make use of information from complete document but in practice, it seems to be not true or at least needs to be further investigated. We therefore need to be cautious with our interpretation of "whole sentence". The seventh column **local?** reports whether the construction of models has a local or global extrema. If it converges to local extrema, it may be useful to combine several models to achieve better results. For example, with *n*-gram class-based LMs, the interpolation of several models with different random clusterings brings some improvements. It should be noted that random forests use it already as they are in fact a sets of many (locally optimized) random decision trees. Finally, the last column **inference complexity** reports the relative measure of each model complexity with three degrees ("fast", "slow" and "very slow").

In this chapter, we examined almost all recent state-of-the-art approaches in language modeling. In spite of having many drawbacks, *n*-gram word based language models remain an essential component in NLP, mainly because of their simplicity that make them scale well with the huge quantity of data which is available nowadays. More sophisticated and promising approaches often suffer from their complexity. In many cases, they are shown to significantly outperform *n*-gram word based approach in small or even medium tasks but their results are discouraged within large scale frameworks.

In summary, there are four main directions. The first one is based on linguistically motivated idea which try to "put back language into language modeling" as claimed by Jelinek (1995), e.g., structured language models, random forest language models . . . The second one aims at using longer contexts, up to the first word of the document such as trigger language models, pLSA or LSA based language models . . . Another direction is to make models more compact by using a more complicated architecture, e.g., exponential or structured language models . . . in order to to tackle the data sparsity problem. Using the similarity between words such as n-gram class-based or similarity based language models is also a possible direction. Among state-of-the-art approaches, continuous space neural networks have been shown to be very promising since it more or less benefits from ideas of all directions[6]. The main problem with this approach concerns the computation complexity. In the next chapters, we will show that by reducing efficiently their complexity, continuous space neural network models can actually learn some useful knowledge to significantly improve system performances in large scale tasks. Furthermore, their achievement is not restricted to language modeling: they will be demonstrated to be also useful for other NLP tasks.

[6]There may be a hesitation for the second direction. In fact, following n-gram assumption but not like n-gram word based language models, NNLMs can take into account longer contexts. As shown in Chapter 5, in practice, the use of 10-gram NNLMs is almost sufficient as context words that are too far seems not to be very helpful.

language model	**feature**	**prob interp?**	**word order**	**word similarity**	**context**	**local?**	**inference complexity**
n-gram word based	words	yes	yes	no	4	no	fast
n-gram class-based	words, classes	yes	yes	hard (classes)	4	yes	fast
structured	words, binary parse tree	yes	yes (flexible)	hard (tags)	whole sentence	yes	very slow
similarity based	words	no	yes	soft (KL divergence)	1	no	slow
topic (pLSA) based	words, topics, documents	yes	no	soft (semantic)	whole document	yes	slow
LSA based	words, documents	no	no	soft (semantic)	whole document	no	slow
random forest	words, tags, prefixes . . .	yes	yes	hard (tags, prefixes . . .)	3	yes	very slow
exponential	words (usual, skip, trigger n-grams)	yes	yes (flexible)	no	whole document	no	very slow
model M	words, classes	yes	yes	hard (classes)	3	no	very slow
feed-forward NN	words	yes	yes	soft (continuous space)	6	yes	very slow
recurrent NN	words	yes	yes	soft (continuous space)	whole sentence (whole document?)	yes	very slow
ranking NN	words	no (score)	yes	soft (continuous space)	5 left + 5 right words	yes	slow

Table 1.2: *Summary for state-of-the-art approaches*

Part II

Continuous Space Neural Network Language Models

Chapter 2

Structured Output Layer

Contents

A major difficulty with the neural network approach is the complexity of inference and training, which mainly depends on the size of the output vocabulary, i.e., of the number of words that have to be predicted. Using the standard feed-forward model presented in Section 1.4 of Chapter 1 makes handling of large vocabularies, consisting of hundreds of thousands of words, unfeasible due to a prohibitive growth in the computation time. To work around this problem, shortlist technique for NNLMs, based on a simplification applied to the output size, was proposed in (Schwenk and Gauvain, 2002). By restricting the output vocabulary to a shortlist, a set that contains several thousands of frequent words, the training and inference are made possible, while a large part of the probabilities for n-grams ending with in-shortlist words can still be estimated by neural network, whereas the other probabilities are estimated with conventional back-off models. For more details on this technique, the reader is referred to Section 1.4.3.2.

To completely overcome this obstacle, we introduced the *Structured OUtput Layer* (SOUL) neural network language modeling approach in (Le et al., 2011b). It is based on a tree representation of the output vocabulary. Following (Morin and Bengio, 2005; Emami, 2006; Mnih and Hinton, 2008), the SOUL model combines the neural network approach with the class-based approach (described in Section 1.3.2). Structuring the output layer and using word class information make the estimation of distribution over vocabularies of arbitrary size computationally feasible. As a result, all the vocabulary words, and not just the words in the shortlist, can benefit from the improved prediction capabilities of the NNLMs.

This chapter provides an overview of the SOUL model along with the main experimental results. A more detailed description and analyses for the SOUL structure is given in Chapter 3. The structure of this chapter is organized as follows. In Section 2.1, we introduce the SOUL architecture. Then, Section 2.2

and Section 2.3 are devoted to sketch out a training algorithm well adapted to this architecture and its enhanced version proposed to better deal with less frequent words. Experimental results both in large scale ASR and SMT tasks with significant improvements are finally reported in Section 2.4.

2.1 SOUL Structure

The main idea leading to the SOUL approach is first motivated by the effort to generalize all approaches which aim to reduce the complexity of NNLMs by using a hierarchical structure for the output vocabulary. From the general structure, the SOUL structure is designed as a tradeoff between the complexity and the performance of models. In this section, we will first present the general form of the hierarchical structure. The SOUL structure will then be introduced.

2.1.1 A general hierarchical structure

Following the n-gram approach, NNLMs aim to predict word w_n given its history w_1^{n-1}. To factorize the conditional word probability, the output vocabulary is structured in a clustering tree, where every word is associated to a unique path from the root node to a leaf node. Let U denote the tree depth, the maximal path length in the clustering tree is therefore $U + 1$. For each path associated to a word in the vocabulary which has number of nodes smaller than $U + 1$, we add artificially a sequence of "false" nodes so as to guarantee that all leaves are at the same depth. The sequence $x_0^U(w_n) = x_0, \ldots, x_U$, where $x_u \in \Pi$ (the set of nodes of the tree), can then be used to encode the path for word w_n in this tree. In this sequence, x_0 is the root of the tree, x_u with $u = 1, \ldots, U - 1$ are the classes assigned to w_n at level u and x_U is the leaf associated with w_n, comprising just the word itself. The probability of w_n given its history w_1^{n-1} can then be computed as:

$$\begin{aligned} P(w_n|w_1^{n-1}) &= P(x_0^U(w_n)|w_1^{n-1}) \\ &= P(x_0|w_1^{n-1}) \times \prod_{u=1}^{U} P(x_u|w_1^{n-1}, x_0^{u-1}) \end{aligned} \tag{2.1}$$

In this equation, the first term can be omitted as it is equal to 1. Each class conditional distribution $P(x_u|w_1^{n-1}, x_0^{u-1})$ is estimated using a *softmax layer* of the neural network, which is associated to node x_{u-1}[1], since we assume a tree structure:

$$P(x_u|w_1^{n-1}, x_0^{u-1} = P(x_u|w_1^{n-1}, x_{u-1}),$$

and the first term w_1^{n-1} is introduced as the input of the neural network.

[1]A neural network now has several softmax layers at the output part.

Note that for "false" nodes, the softmax layer is not required as they are single children, hence, the conditional probabilities of these nodes are all 1. These nodes are only introduced in order to simplify the description for the general hierarchical structure. As defined in Section 1.4.1.1, a softmax layer takes as input the hidden layer and computes the values of the output layer, then producing a normalized distribution after applying the softmax function.

If another false node x_{U+1} is added as the last node for all paths in the tree, we can see that the factorization in Equation (2.1) for a node sequence is similar to the one used in Equation (1.1) for a word string. x_0 has the same role as <s>, likewise for x_{U+1} and </s>. The node vocabulary is Π. The main difference is that the clustering tree defines a restricted amount of possible node sequences. Therefore, the conditional probabilities for (sub)sequences of nodes which cannot be induced from the tree is null. In NNLMs, the others are estimated using normalized distributions provided by softmax layers. Because Equation (1.1) well reflects a normal distribution, it can easily be shown that by using Equation (2.1), the output distributions estimated by NNLMs also satisfy the sum to one condition.

As described in Section 1.3.7, exponential language models have the same computational problem of the softmax in the output layer and hence, a solution based on clustering words into classes was also proposed in (Goodman, 2001a). In this approach, two models are trained separately: one that predicts the class probability given the history; and one that predicts a word given its class and its history. A significant speed-up can be obtained since the required computations for both models are drastically reduced: for the first model, the summation implied in the normalization ranges over the number of different classes, whereas for the second model it involves only the words in the given class.

The idea of clustering the vocabulary words was introduced for NNLMs in (Morin and Bengio, 2005), where the author tried to derive a binary hierarchical clustering from the WordNet semantic hierarchy. Alternative solutions that do not rely on external linguistic resources were proposed in (Emami, 2006; Mnih and Hinton, 2008) for Log-BiLinear (LBL) models. This type of model, called Hierarchical Log-BiLinear (HLBL), has been described in detail in Section 1.4.1.3. The output vocabulary is clustered and represented as a binary tree. Each internal node of the tree holds a word cluster which is further divided into two subclusters and so on, until the leaves, each of which corresponds to a vocabulary word. This approach results in a two order of magnitude speed-up on a small data set (1 million words of the Brown corpus) with a vocabulary of 10k words. However, a significant increase in perplexity is observed relative to a standard NNLM. Because each "sigmoid" node of the output layer is equivalent to a softmax layer of 2 nodes, the model structure in this approach is in fact a particular case of our more general proposal.

Factorization of the output layer was also used for RNNLMs (see Section 1.4.1.4 for more details). Proposed in (Mikolov et al., 2011c), this approach does not use any relationships between words since it is based on the distribution of unigram probabilities. Another active direction of research dealing with different clustering methods is connected to the recently proposed Model M and their subsequent enhancements (Chen and Chu, 2010; Emami and Chen, 2011) (see Section 1.3.8).

2.1.2 A SOUL structure

On the one hand, it can be seen that much of the previous work focused on reducing the computational complexity, with less emphasis on improving the accuracy of speech recognition or machine translation systems. Particularly, in the HLBL approach, the resulting binary clustering tree is deep (i.e., has many levels), and thus faces a data fragmentation problem. This problem is well known for decision trees, especially for language modeling tasks (Potamianos and Jelinek, 1998; Xu and Jelinek, 2007). Moreover, if a word is assigned to a wrong class at some level in a decision tree, this error affects all the internal nodes (or clusters) leading to this word. This is typically the case for rare words that represent a large fraction of the vocabulary. Thus an error in one word may have a significant impact on the whole system. So in the SOUL structure, by relaxing the constraint of the binary structure to make the tree shallower, it is expected that this shortcoming will be overcome.

On the other hand, as shortlist based NNLMs (see Section 1.4.3.2) have been shown to consistently achieve state-of-the-art performances, it is intuitively guaranteed that we will not have any degradation on the system performance if we grow the SOUL structure from the shortlist one. As a result, in the SOUL structure, we propose to keep the in-shortlist (most frequent) words special, as they are not clustered, each of them represents a class on its own at the first level. Only the remaining words, called out-of-shortlist (OOS) words, are first clustered into top classes (at the first level, the same level as the in-shortlist words), then subclasses (at the deeper levels).

An example of the SOUL architecture is displayed in the right part of Figure 2.1. The output part of the model is made of two main parts:

1. *The main softmax layer* estimates $P(x_1|w_1^{n-1})$, the class probability and the probability of in-shortlist words;

2. *The remaining softmax layers* estimate $P(x_u|w_1^{n-1}, x_0^{u-1}), u = 2 \dots (U)$, the subclass probabilities;

Note that above, we only provide an overview. For deep investigations on the SOUL architecture design, the reader is invited to go to Chapter 3.

2.2 Training algorithm

Given a clustering tree, a SOUL NNLM can be trained using the stochastic gradient descend (SGD) algorithm as is done for standard feed-forward NNLMs

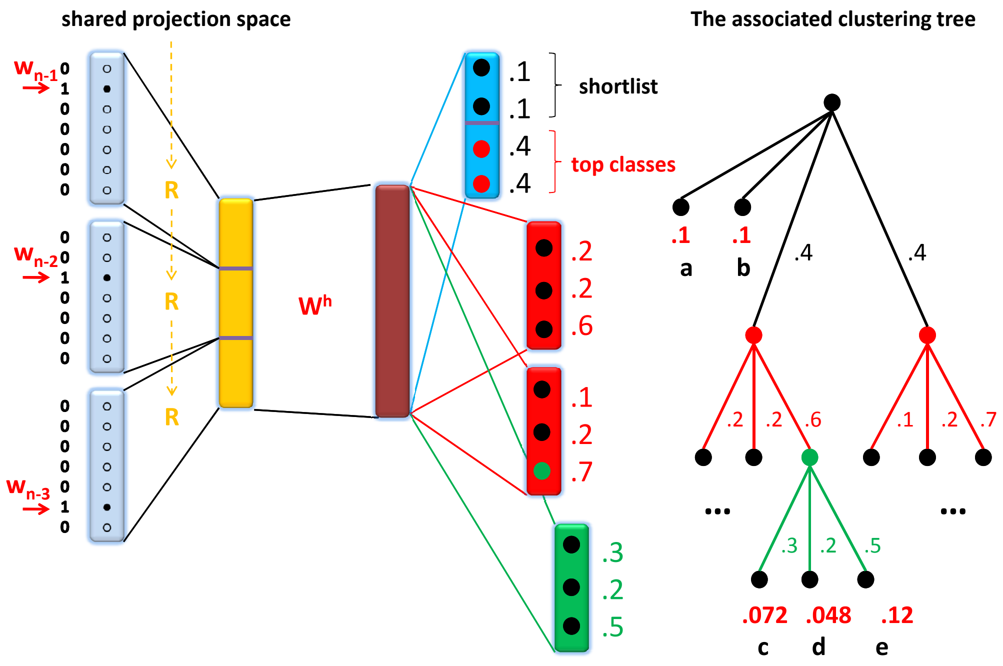


Figure 2.1: *SOUL neural network language model architecture. In this example, the main softmax layer has two parts: one for the shortlist of* 2 *words (**a** and **b**) and one for* 2 *top classes for the other words. At the level* 2*, each top class is divided into* 3 *subclasses. Three words **c**, **d** and **e** have their associated leaves at the level* 3*. To compute the probability for each word, we make a product over the probabilities in the path (from the root to its leaf), e.g., the probability assigned to **e** is* $0.12 = 0.4 \times 0.6 \times 0.5$*; to **a** is* $0.1 = 0.1$*.*

(see Section 1.4.2). The presence of several softmax layers at the output side of SOUL NNLMs only leads to a little modification of the derivatives and of the equations, as shown in Appendix C. In the original article, in order to obtain the clustering tree from an estimate of the continuous space, we proposed a sophisticated training scheme for SOUL NNLMs that can now be summarized in the three main steps of the Algorithm 2. Note that the word clustering tree can also be obtained from other algorithms, so the reader is referred to Section 3.1 of Chapter 3 for the comparison between three clustering algorithms: the SOUL clustering, the Brown clustering (Brown et al., 1992) and the unigram clustering (Mikolov et al., 2011c).

Algorithm 2 SOUL training scheme

```
**Step 1, shortlist pre-training** provides a first estimate of the continuous space by training a shortlist NNLM.
**Step 2, word clustering** derives a clustering tree from the information in the continuous space.
**Step 3, full training** trains a SOUL NNLM with the tree structure induced in Step 2 and with the parameters initialized using the parameters obtained at Step 1.
```

We outline here the way we carry out each step. The reader is encouraged to go to Chapter 3 for more analyses on the SOUL configuration, e.g., the clustering algorithm described in detail along with its hyper-parameters, the size of the shortlist, the number of top classes ...

At Step 1 : To create a first estimate of the continuous space, we train an NNLM with a shortlist comprising the most frequent words as the output vocabulary. As only a sufficiently good word space is required, the model is trained for a few iterations, not necessarily until convergence.

At Step 2 : As words input to an NNLM are represented in 1-of-V coding with 1 corresponding to a word's vocabulary index and all other elements set to zero, each line in the projection matrix $\mathbf{R}$ corresponds to a continuous representation of a particular word. A recursive clustering algorithm is therefore applied to the representations of OOS words in this continuous space in order to obtain a hierarchical structure.

At Step 3 : By adding the part for OOS words induced at Step 2 to the output part of the model obtained at Step 1, a SOUL NNLM with a whole vocabulary on the output side is created. Initializing with the parameters obtained at Step 1, this model is then trained until convergence following the classical SGD algorithm.

2.3 Enhanced training algorithm

A new training scheme for the class part of SOUL NNLMs used to deal with OOS words was proposed in (Le et al., 2011a). It makes a better use of available data during training, trying to provide more robust parameter estimates for large vocabulary tasks. Application of this scheme results in additional improvements in perplexity and system performance.

The main reason for proposing this new scheme comes from the limitation of resampling techniques, described in Section 1.4.3.2. Resampling of training data is conventionally used as it is computationally unfeasible to train an NNLM on the same amount of data as is used a conventional *n*-gram language models. Usually, at each epoch, training examples up to several million words are randomly selected. When dealing with large vocabularies, the number of parameters at the output part of SOUL NNLMs is much larger than that of shortlist based NNLMs. As a result, using the same number of resampled examples as for shortlist based NNLMs to train SOUL NNLMs may be insufficient to obtain robust parameter estimates.

To make it clearer, let us consider again the output part of SOUL NNLMs. It comprises of two parts: the first one which contains the main softmax layer which directly models the probabilities of the most frequent in-shortlist words and top classes for OOS words; the second one which is composed of the remaining softmax layers used to deal with OOS words as displayed in Figure 2.1. The parameters related to the first part are updated for all training examples since they cover the most frequent (in-shortlist) words and the top (most general) classes for the less frequent (OOS) words. The *n*-grams ending with in-shortlist words are used to update the parameters only of the main softmax layer, leaving the other layers intact. The parameters of the other layers are updated with the *n*-grams ending in an OOS word and only those layers leading to the leaf with this particular word are activated and the corresponding parameters updated. A shortlist usually covers a large part of training examples so that the updates of the parameters related to the second part are less frequent. Moreover, when such an update occurs, it is only performed for a small subset of the parameters corresponding to a particular path in the clustering tree. At the same time, the number of parameters of this second part is much larger[2]. As a result, the two parts of the SOUL output layer are not equally well trained.

A modified SOUL training scheme (Algorithm 3) based on the separate training of the OOS part is therefore proposed. This scheme adds an additional step to the training procedure. This additional Step 3 is similar to Step 1, but is carried out only for OOS words. At this step the *n*-grams ending with shortlist words are skipped and the parameters associated with shortlist words are

[2]If we denote the number of shortlist words by $\mathcal{V}^s$, the number of top classes by $\mathcal{V}^{x_1}$, the number of parameters of the first part is $(H+1) \times (\mathcal{V}^s + \mathcal{V}^{x_1})$ and of the second parts is larger than $(H+1) \times (\mathcal{V} - \mathcal{V}^s - \mathcal{V}^{x_1})$.

Algorithm 3 Enhanced SOUL training scheme

```
Step 1, shortlist pre-training provides a first estimate of the continuous space by
training a shortlist NNLM.
Step 2, word clustering derives a clustering tree from the information in the contin-
uous space.
Step 3, OOS pre-training trains an NNLM with OOS words as output.
Step 4, full training trains a SOUL NNLM with the tree structure induced in Step
2 and with the parameters initialized using the parameters obtained at Step 3 and
Step 1.
```

temporarily fixed, reducing the size of the main softmax layer to the number of top classes only. As explained above, the softmax in this layer is triggered for each training example, which, in turn, requires summation over all nodes in this layer. Thus, any reduction in the size of the main softmax layer results in a significant speed-up. Since the number of OOS occurrences represents only a relatively small proportion of the training data, the number of training examples processed through this layer can be easily increased by the factor of 10. Finally, it should be noted that the parameters obtained at Step 3 are used to initialize the parameters associated with the OOS words at Step 4 in the same way as the parameters obtained at Step 1 are used for the shortlist part of the main softmax layer.

2.4 Experimental evaluation

The advantages of new proposed SOUL NNLMs have been empirically demonstrated in various contexts, both for ASR and SMT large scale tasks with different languages (Mandarin, Arabic, French, English …). In this section, we summarize our several recent experimental results reported in several publications (Le et al., 2011b; Le et al., 2011a; Allauzen et al., 2011).

2.4.1 Automatic Speech Recognition

In this section, Mandarin and Arabic data are used to evaluate the SOUL NNLM accuracy via ASR experiments. Well-tuned ASR systems developed for the GALE program (with error rates around 10%) serve as the very strong baselines (see (Oparin, Lamel, and Gauvain, 2010a) for the Mandarin configuration and (Lamel, Messaoudi, and Gauvain, 2009) for the Arabic one). In the configurations, even an absolute improvement of 0.1% WER has been shown to be very hard to achieve. Moreover, with training data of about several billion words, most state-of-the-art approaches to language modeling (except shortlist NNLMs) fail to bring improvements over robust conventional n-gram word based language models. Via lattice rescoring, the SOUL models are shown to achieve consistent improvements over classical shortlist NNLMs both in terms of perplexity and recognition accuracy for these two languages, even though that are quite different in terms of their internal structure and recognition vocabulary size. An enhanced training scheme is shown to achieve additional improvements in the Arabic task.

2.4.1.1 ASR Setup

Mandarin To segment Mandarin phrases in words, we make use of the simple longest-match segmentation algorithm based on 56,052 word vocabulary used in previous LIMSI Mandarin ASR systems (Luo, Lamel, and Gauvain, 2009). However, character error rate (CER) is conventionally used to evaluate final recognition performance for Mandarin.

The GALE *dev09* and *eval09* sets are used in this study to evaluate the performance of different models. This data consists of broadcast news and broadcast conversations. A subset of *dev09* called *dev09s* was also defined. It constitutes about a half of *dev09* data. More details concerning the experimental setup, acoustic models and decoding process of the baseline LIMSI Mandarin ASR system can be found in (Oparin, Lamel, and Gauvain, 2010a).

The first part language model is trained on 3.2 billion word tokens (after segmentation) of Mandarin data thus providing the system with a robust LM. The baseline LM is a word-based 4-gram LM. Individual LMs are first built for each of the 48 Mandarin corpora that are available by the end of the year 2009. These models are smoothed according to the interpolated Kneser-Ney discount scheme. No cut-offs nor pruning are applied thus making the LMs to take into account all the available information. These individual models are subsequently linearly interpolated with interpolation weights tuned on *dev09* data. As the number of individual models is not very large, this particular choice of a training set does not result in a bias to this data.

For shortlist based neural networks, we follow the algorithm presented in (Schwenk, 2007) with two configurations for the size of the shortlist (8k and 12k). For SOUL NNLMs, we follow exactly the 3 step algorithm as presented in Section 2.2. For each epoch, each NNLM is trained on about 25M words after resampling of the training data. For each test configuration, four NNLMs of the same type are trained and interpolated. They differ in the dimension of the shared context space, size of the hidden layer and training data resampling rate. For all our experiments, the learning rate of different NNLMs is 5×10^{-3}, the learning weight decay is 5×10^{-8} and the weight decay is 3×10^{-5}. The configuration parameters are summarized in Table 2.1.

State-of-the-art n-gram language models are rarely of an order larger than 4. Our previous experiments on very large setups indicated that the gain obtained when increasing the n-gram order from 4 to 5 is almost negligible while the size of models increases drastically. Handling such models is thus quite impractical and can hardly be done without pruning. However, this is not the case for NNLMs, due to the different nature of these models. The increase in context length at the input layer results in only at most linear growth in complexity (Schwenk, 2007). Thus training NNLM with longer

parameter	shortlist based NNLM	SOUL NNLM
training data size	$\approx 3200M$	
vocabulary size	56,052	
number of models	4	
order (n)	4 or 6	
projection space (M)	$300, 250, 200, 220$	
hidden layer size (H)	$500, 450, 500, 430$	
nonlinear activation	tanh	
shortlist size	$8k$ or $12k$	$8k$
number of top classes	-	$4k$
number of examples/epoch	$\approx 25M$	
number of epochs (shortlist)	≈ 10	3
number of epochs (all words)	-	≈ 10
learning rate	5×10^{-3}	
learning rate decay	5×10^{-8}	
weight decay	3×10^{-5}	
block size	128	

Table 2.1: *Mandarin: NNLM configurations.*

contexts is still feasible. As our aim is to improve the performance of the Mandarin ASR system, we also investigated the increase in context length from 3 (that corresponds to 4-grams) to 5 for different NNLMs, while keeping the same 4-gram back-off LM at the output layer for standard shortlist NNLMs. Combining models of different order remains a valid thing to do with the back-off scheme used for NNLMs with shortlists, described in Section 1.4.3.2.

Arabic A challenging Arabic GALE task characterized by a vocabulary of about $300k$ entries is chosen to evaluate the SOUL NNLM performance for large vocabularies. State-of-the-art LIMSI Arabic ASR system (presented in (Lamel, Messaoudi, and Gauvain, 2009)) is used to perform speech recognition experiments.

Arabic is a highly inflective and morphologically rich language. In order to deal with its peculiarities, decomposition of words in its morphological constituents was shown to improve speech recognition results (Lamel, Messaoudi, and Gauvain, 2008; Diehl et al., 2009). There are several approaches to Arabic word decomposition. In this study we use one of the most popular tools up-to-date, namely MADA: Morphological Analysis and Disambiguation for Arabic (Roth et al., 2008).

Language model training data consist of about 1.7 billion words before morphological decomposition. The recognition vocabulary contains 296,772 entries. Altogether, 32 interpolated Kneser-Ney 4-gram LMs for different text corpora are trained on MADA-decomposed units (about 2 billion in total) without pruning nor cut-offs. These models are further interpolated to create the final LM that serves as a robust baseline model. Lattices generated with this model are subsequently rescored with NNLMs.

Three NNLMs are trained with different resampling parameters, sizes of context $(200, 300, 400)$ and hidden layers $(500, 400, 300)$ for each n-gram order. These models are interpolated in order to obtain final NNLMs. Resampling is performed with a bias towards using corpora containing broadcast news (*bn*) and broadcast conversations (*bc*) data as target data. Up to $30M$ words data are used at each NNLM iteration of shortlist-based and standard SOUL NNLMs. For enhanced SOUL NNLMs, the resampling rate is augmented 10 times at Step 3. It results in about $30M$ n-gram examples used to train the output part that deals with OOS words. All training parameters for the SOUL NNLMs are kept the same as for the shortlist NNLMs. Thus it can be argued that the difference in performance is due to the use of the whole vocabulary at the output layer. These configurations are summarized in Table 2.2.

parameter	**shortlist based NNLM**	**SOUL NNLM**
training data size	$\approx 2000M$	
vocabulary size	296,772	
number of models	3	
order (n)	4 or 6	
projection space (M)	$200, 300, 400$	
hidden layer size (H)	$500, 400, 300$	
nonlinear activation	tanh	
shortlist size	$8k$ or $12k$	$8k$
number of top classes	-	$4k$
number of examples/epoch	$\approx 30M$	
number of epochs (shortlist)	≈ 15	5
number of epochs (OOS)	-	10
number of epochs (all words)	-	≈ 15
learning rate	5×10^{-3}	
learning rate decay	5×10^{-8}	
weight decay	3×10^{-5}	
block size	128	

Table 2.2: *Arabic: NNLM configurations.*

Three GALE validation and evaluation sets are used to evaluate the performance of different models, namely *dev09s*, *eval10ns* and *dev10c*. These sets consist of 23,576, 45,629 and 52,181 MADA decomposed units respectively.

2.4.1.2 Results

Mandarin Perplexity and recognition accuracy for different models are reported in Table 2.3. Perplexity results are on *dev09* and CER results are on *dev09s* and *eval09*.

The row +4-gram $8k$ shortlist NNLM corresponds to the results when the baseline 4-gram model is interpolated with the baseline NNLMs that make use of a shortlist of $8k$ most frequent words and take into account a context of the same length. The row +6-gram $8k$ shortlist NNLM reports results related to the interpolation of the baseline 4-gram LM with the longer context NNLMs. Training a baseline 6-gram LM on 3.2 billion words is unfeasible without severe cut-offs and, according to our experience with 5-gram LMs, only minor improvements over 4-grams can be achieved.

We try the longer context neural network setup with the shortlist increased up to 12k most frequent words. Results obtained with this setup are reported in the rows +4-gram 12k shortlist NNLM and +6-gram 12k shortlist NNLM in Table 2.3. Finally, results obtained with the whole-vocabulary SOUL NNLMs are represented in rows 4 and 7.

model	**ppl**	**CER**	
	dev09	*dev09s*	*eval09*
4-gram baseline	211	9.8	8.9
+4-gram 8k shortlist NNLM	187	9.5	8.6
+4-gram 12k shortlist NNLM	185	9.4	8.6
+4-gram SOUL NNLM	180	9.3	8.5
+6-gram 8k shortlist NNLM	177	9.4	8.5
+6-gram 12k shortlist NNLM	172	9.3	8.5
+6-gram SOUL NNLM	**162**	**9.1**	**8.3**

Table 2.3: *Mandarin: Perplexity and CER (in %) for language models.*

The results presented in this study suggest several conclusions. Contrary to classical back-off n-gram LMs, increasing the NNLM context length significantly improves the results both in terms of perplexity and CER, without any major impact on the training and probability computation time. This is true for all NNLMs, which all improve the Kneser-Ney baseline LM trained on large amounts of data. This again shows the capability of NNLMs to better overcome data sparsity issues.

The gains achieved with SOUL NNLMs correspond to a relative improvement of 23% in perplexity and $7-9\%$ in CER. SOUL NNLMs also outperform shortlist NNLMs due to the fact they predict all words from the vocabulary (as other parameters are kept the same). The most significant improvement with SOUL models is obtained for the longer context (6-gram) NNLMs configuration.

Arabic Perplexity and WER results are presented in Tables 2.4 and 2.5 respectively. The first row in the tables corresponds to the 4-gram baseline Kneser-Ney LM. For the shortlist-based NNLMs, 12k most frequent words form the shortlist. Six-gram NNLMs are trained in order to verify possible improvements from using longer context as opposed to the usual 4-gram setup.

Perplexity results in Table 2.4 are given both when using only NNLMs (columns *alone*) and for the cases NNLMs are interpolated with the baseline 4-gram LM (columns *int.*). Models marked as “SOUL” are conventional SOUL

NNLMs, while "SOUL+" corresponds to the SOUL NNLMs that make use of the enhanced training scheme (Algorithm 3). The latter uses more data to train the OOS part of output layers.

Table 2.4: *Arabic: Perplexity for different language models.*

LM type	**dev09s**		**eval10ns**		**dev10c**	
	alone	*int.*	*alone*	*int.*	*alone*	*int.*
4-gram baseline	312		239		256	
4-gram 12k shortlist NNLM	324	276	247	213	256	224
4-gram SOUL NNLM	293	256	225	200	231	208
4-gram SOUL+ NNLM	277	250	214	195	221	204
6-gram 12k shortlist NNLM	302	263	228	202	236	210
6-gram SOUL NNLM	255	231	196	180	200	186
6-gram SOUL+ NNLM	**245**	**227**	**189**	**177**	**194**	**183**

Using longer context brings improvements both for shortlist and SOUL NNLMs. The ability of neural network LMs to improve performance with the increase of context goes in line with results obtained with RNNLMs (see Section 1.4.1.4). The latter can be regarded as a special case of neural networks taking into account all the history seen before the predicted word. Further comparison with RNNLMs are carried out in Chapter 5.

Perplexity results show that SOUL NNLMs (both 4-gram and 6-gram) outperform the baseline 4-gram LM trained on much bigger data. SOUL NNLMs also consistently outperform shortlist NNLMs of the same orders on all the test sets. Relative improvements of about 10% for stand-alone NNLMs and 7% for interpolated models are observed for 4-gram NNLMs on different test sets. For longer context 6-gram NNLMs, the gains from using SOUL NNLMs are even larger, about 15% and 12% in alone and interpolated scenarios respectively.

Only minor gains in perplexity are achieved with the enhanced SOUL NNLM training scheme as compared to standard SOUL NNLMs. This suggests that using 10 times more data to train OOS part of SOUL NNLMs does not have much influence on model performances in this task.

Table 2.5: *Arabic: WER (in %) for different language models.*

LM type	**dev09s**	**eval10ns**	**dev10c**
4-gram baseline	14.8	9.6	14.5
+ 4-gram 12k shortlist NNLM	14.4	9.1	14.2
+ 4-gram SOUL NNLM	14.3	9.0	14.0
+ 4-gram SOUL+ NNLM	14.1	9.1	14.0
+ 6-gram 12k shortlist NNLM	14.3	9.1	14.2
+ 6-gram SOUL NNLM	14.0	8.9	14.0
+ 6-gram SOUL+ NNLM	14.0	8.9	13.9

Results in Table 2.5 show that the improvements in perplexity obtained with SOUL NNLMs over shortlist NNLMs carry over to speech recognition experiments. The interpolation weights for language models are tuned on GALE Phase 5 validation data and are given in Table 2.6. They show that the SOUL NNLMs obtain higher interpolation weights as compared to the shortlist NNLMs.

As can be seen in Table 2.5, using SOUL NNLMs results in better recognition performances as compared to shortlist NNLMs both for 4-gram and 6-gram cases. The gains from using 6-gram NNLMs are smaller than could be expected as the lattices have to be pruned before rescoring with 6-grams for computational reasons. The effect of pruning is most notable on *dev10c* set. This set contains some large lattices that need more severe pruning. However, as 6-gram shortlist NNLMs show no improvement with pruned lattices over 4-gram NNLMs, 6-gram SOUL NNLMs still improve the results.

Because the Arabic vocabulary is 6 times larger than the Mandarin one, larger gains could be expected from using SOUL NNLMs, as they estimate probabilities for all n-grams and not only for those ending with an in-shortlist word. However, similar gains are observed. This might be due to the fact that in both cases, a relatively high coverage is achieved with shortlists. Shortlists are formed on the basis of all training data before resampling. For 12k shortlists used in Mandarin and Arabic NNLMs, coverage on the basis of all training data is 95% and 90% respectively. The overall coverage shows that though Arabic vocabulary is several times larger than the Mandarin one, the shortlist coverage is only 5% lower. As training data is resampled at each epoch with the emphasis on sources containing target *bn* and *bc* data, the number of calls to an NNLM (i.e., coverage at each epoch) changes. The same is valid for validation data due to the fact it consists only of *bn* and *bc* data and the general vocabulary may have different coverage. We check the coverage on validation and test data and observe no significant difference as compared to the overall coverage. The shortlist coverage statistics shows that similar size shortlists do relatively well in terms of data coverage even for models with much larger vocabularies.

Finally, enhanced SOUL NNLMs bring slight improvements over standard SOUL NNLMs on some data sets and NNLM configurations (4-gram NNLM on *dev09s* and 6-gram NNLM on *dev10c*).

Table 2.6: *Arabic: The weights for neural network language model when interpolated with the baseline n-gram model.*

LM type	**interpolation weight**
4-gram 12k shortlist NNLM	0.50
4-gram SOUL NNLM	0.68
4-gram SOUL+ NNLM	0.72
6-gram 12k shortlist NNLM	0.55
6-gram SOUL NNLM	0.74
6-gram SOUL+ NNLM	0.75

2.4.2 Machine Translation

Results presented in this section are taken from the LIMSI submissions to the WMT 2011 English to French and English to German shared translation tasks (Allauzen et al., 2011). The baseline system is built using *n-code*, our open source Statistical Machine Translation system based on bilingual n-grams (Crego, Yvon, and Mariño, 2011). The use of SOUL NNLMs of large order to rescore *m*-best lists provided by baseline systems has been shown to achieve significant and consistent improvements.

2.4.2.1 MT Setup

We only provide here a very short overview of the task; all the necessary details regarding this evaluation campaign are available on the official Web site[3]. Simply note that our parallel training data includes a large Web corpus, referred to as the *GigaWord parallel corpus*. After various preprocessing and filtering steps, the total amount of training data is approximately 12 million sentence pairs for the bilingual part, and about 2.5 billion of words for the monolingual part.

To create a baseline conventional language model, we assume that the test set consists in a selection of news texts from the end of 2010 to the beginning of 2011. This assumption is based on what was done for the 2010 evaluation. Thus, for each language, we build a development corpus in order to optimize the vocabulary and the target language model.

To estimate such large baseline n-gram LMs, a vocabulary is first defined for each language by including all tokens observed in the Europarl and News-Commentary corpora. For French, this vocabulary is then expanded with all words that occur more than 5 times in the French-English *GigaWord* corpus, and with the most frequent proper names taken from the monolingual news data of 2010 and 2011. As for German, since the amount of training data is smaller, the vocabulary is expanded with the most frequent words observed in the monolingual news data of 2010 and 2011. This procedure results in a vocabulary containing around 500k words in each language.

For French, all the training data allowed in the constrained task are divided into 7 sets based on dates or genres. On each set, a standard 4-gram LM is estimated from the 500k words vocabulary using absolute discounting interpolated with lower order models (Kneser and Ney, 1995; Chen and Goodman, 1998). All LMs except the one trained on the news corpora from 2010-2011 are first linearly interpolated. The associated coefficients are estimated so as to minimize the perplexity evaluated on *dev2010-2011*. The resulting LM and the 2010-2011 LM are finally interpolated with *newstest2008* as validation data. This procedure aims to avoid overestimating the weight associated to the 2010-2011 LM.

[3] http://www.statmt.org/wmt11

For German, as we have a smaller quantity of data, all the training data are concatenated. From that, one LM is trained using a Kneser-Ney smoothing technique.

For SOUL NNLMs, we train four interpolated models that use a hidden layer of 300 units and a projection word space of 400 dimensions. The 5000 most frequent words form a shortlist. They differ in the resampling rates that prefer different parts of data. As mentioned erlier, the order of a continuous n-gram model such as SOUL NNLMs can be increased at a small (linear) computational cost. We therefore try different configurations (4, 6 and 10-gram) in order to verify the benefit of using larger orders for SOUL NNLMs. In Table 2.7, we show the chosen configuration parameters in more details[4]

SOUL LM scores are introduced as a new score in the SMT pipeline by rescoring the m-best list generated by the decoder, and the associated weight was tuned with MERT algorithm (Och, 2003).

parameter	**SOUL NNLM**
training data size	$\approx 2500M$ (French); $\approx 360M$ (German)
vocabulary size	$\approx$ 500,000
number of models	4
order (n)	4, 6 or 10
projection space (M)	400
hidden layer size (H)	300
nonlinear activation	sigmoid
shortlist size	5000
number of top classes	5000
number of examples/epoch	$\approx 50M$
number of epochs (shortlist)	3
number of epochs (OOS)	5
number of epochs (all words)	15
learning rate	1×10^{-2}
learning rate decay	not used
weight decay	3×10^{-5}
block size	32

Table 2.7: *WMT 2011: NNLM configurations.*

2.4.2.2 Results

The experimental results are reported in terms of BLEU and TER using the *newstest2010* corpus as evaluation set. These automatic metrics are computed using the scripts provided by NIST after a detokenization step. We summarize in Table 2.8 our experiments with SOUL NNLMs of orders 4, 6, and 10.

[4]The learning rate is larger than the one used in the previous ASR experiments because we use another nonlinear activation function, sigmoid in lieu of tanh. The learning rate decay is marked *not used* because we follow the second training regime presented in Section 1.4.2.2.

We observe for the English-French task: a BLEU improvement of 0.3, as well as a similar trend in TER, when introducing 4-gram SOUL NNLMs; an additional BLEU improvement of 0.3 when increasing the order from 4 to 6; and a less important gain with the 10-gram SOUL NNLM. In the end, the use of a 10-gram SOUL NNLMs achieves a 0.7 BLEU improvement and a TER decrease of 0.8. The results on the English-German task show the same trend with a 0.5 BLEU point improvement.

model	**en2fr**		**en2de**	
	BLEU	*TER*	*BLEU*	*TER*
baseline	28.1	56.0	16.3	66.0
+ 4-gram SOUL NNLM	28.4	55.5	16.5	64.9
+ 6-gram SOUL NNLM	28.7	55.3	16.7	64.9
+ 10-gram SOUL NNLM	28.8	55.2	16.8	64.6

Table 2.8: *WMT 2011: Translation results from English to French (**en2fr**) and English to German (**en2de**) measured on newstest2010 with SOUL NNLMs.*

2.5 Summary

The SOUL neural network approach to language modeling combines two techniques that were proved to improve both ASR and SMT system performances for large-scale tasks, namely neural network and class-based language models. This approach allows us to train neural network LMs with full vocabularies without confining their power to predicting words from limited shortlists.

Large scale and well-tuned GALE Mandarin and Arabic ASR systems with error rate of about 10% are chosen to evaluate SOUL NNLMs. In these systems, robust baseline language models are trained on very large data, of several billion words. In a GALE Mandarin ASR system, SOUL NNLMs outperform standard shortlist NNLMs for all test configurations. The performance of the SOUL structure on a larger vocabulary containing approximately 300k entries is studied using Arabic GALE task. As for Mandarin, the results on Arabic show that SOUL NNLMs consistently outperform conventional shortlist NNLMs both in terms of perplexity (up to 15% relative improvement) and recognition error rate (up to 0.3% absolute improvement). Despite the fact that the Arabic vocabulary is 6 times larger than the Mandarin one, using SOUL NNLMs bring only similar gains. This suggests that improvements from using full vocabulary SOUL NNLMs, while being consistent, are not proportional to the vocabulary size.

An enhanced training scheme is evaluated using this Arabic ASR task. The new method assumes separate training of the part related to in-shortlist words (as each of these forms a separate class itself without subclustering) and the class part that deals with all other words from the vocabulary. One order of magnitude more data is used to train the OOS part of SOUL NNLMs without any drastic increase in computational charge and training time. However, only minor improvements in perplexity and WER are observed.

To conclude, consistent perplexity and speech recognition improvements over both conventional n-gram baselines and shortlist NNLMs on the GALE Mandarin and Arabic STT tasks make the SOUL NNLM an alternative to the shortlist approach to neural network language modeling. The application of the SOUL NNLM is not confined to speech recognition but can be used for other language technology tasks. For instance, the conclusions drawn from our machine translation results for the WMT 2011 evaluation are along in the same lines as those from ASR evaluations.

Chapter 3

Setting up a SOUL network

Contents

In the previous chapter, a novel output structure for neural network language models, SOUL, have been presented. Though this new approach has been shown to significantly outperform the NNLMs using shortlist at the output, the impact of its various configuration parameter remains to be analyzed. This chapter is thus meant to study the SOUL configuration in more details. Most of the work here is published in (Le et al., 2013).

We first compare different ways of performing word clustering in order to show that our method based on the context word space induces a good hierarchical structure at a small computational cost. We then investigate the impact of different configuration parameters of a SOUL tree clustering, such as the size of the shortlist, the number of top classes in the main softmax layer and the depth of the tree. The main goal is to demonstrate that, following the enhanced SOUL training scheme, the configuration of the output part can be adjusted to significantly reduce the computational time without any degradation of the system performance.

Finally, we relate and discuss our attempts to introduce *Deep learning* into the SOUL NNLM approach. Deep learning is one of the most active research areas in Machine Learning nowadays. It aims at modeling complex relationships by extracting multiple levels of representation from the data. Inspired by the architecture of the human brain which processes information in a hierarchical fashion with multiple levels of data representations and transformations, the concept of deep neural networks having several hidden layers was introduced (Utgoff and Stracuzzi, 2002; Bengio and LeCun, 2007). However, due to some practical training issues, *deep* architectures were often outperformed by *shallow* ones with a smaller number of hidden layers. The reader is referred to (Bengio, 2009) for more details about the deep and shallow architectures. The introduction of deep learning architectures based on Deep Belief Networks with a more sophisticated learning algorithm (Hinton, Osindero, and Teh, 2006) can be considered as a breakthrough. The main idea is to pre-train the network on a layer by layer basis in an unsupervised way, using Restricted Bolzmann Machines (RBM) in order to increase the generalization capacity. Deep learning has been recently applied with success in many fields, especially in the domain of image processing (Hinton, Osindero, and Teh, 2006) or recently in acoustic modeling (Seide, Li, and Yu, 2011; Mohamed, Dahl, and Hinton, 2012) and in NNLMs (Arisoy et al., 2012), albeit within a small scale framework. Using another hidden layer in NNLMs was proposed in (Mikolov et al., 2011b) under the name of a "compression layer". The goal of this layer is to decrease the size of the second hidden layer that is directly connected to the output softmax layers and hence to reduce the computational time rather than to improve the overall accuracy. Following these ideas, the SOUL NNLM structure has been modified to include additional layers. The deeper structure will be shown to bring additional improvements.

The remaining of this chapter is organized as follows. In Section 3.1, we make a comparison between several ways to perform word clustering. Then, Section 3.2 is devoted to investigate various configurations of the tree structure at the output side of SOUL NNLMs. Models with more than one hidden layer are finally examined in Section 3.3. Note that all experiments are carried out on the same Mandarin setup as in Section 2.4.1.1, they all rely on an enhanced SOUL training scheme presented earlier (refer to Algorithm 3 in Section 2.3).

3.1 Word clustering algorithms

In the ASR experiments presented in the previous chapter, at Step 2 of both Algorithm 2 and Algorithm 3, the hierarchical word clustering of SOUL NNLMs was automatically derived from the continuous word space obtained at Step 1. This word clustering algorithm, referred to as NNLM, is described in more details as follows.

At Step 1, to create a first estimate of the continuous space, we train an NNLM with a shortlist comprising the $8k$ most frequent words as the output vocabulary, using the one vector initialization method introduced in (Le et al., 2010) (see Section 4.1.3.3). Only a few iterations (3) are required as the learning of word features converges quickly.

At Step 2, as words input to an NNLM are represented in 1-of-V coding with 1 corresponding to a word's vocabulary index and all other elements set to zero, each line in the projection matrix $\mathbf{R}$ corresponds to a continuous representation of a particular word. Our word clustering method is then applied to the representations of less frequent words in the context space represented by matrix $\mathbf{R}$. This method is based on the relationship between the two word spaces defined in the standard NNLM (Le et al., 2010): the context (input) and prediction (output) spaces (see Chapter 4 for an analysis of this relationship). As we cluster only less frequent words according to their prediction role, ideally we have to use the prediction space at Step 2. In practice, the context space is used instead because the relationships encoded in the context space for less frequent words have been reported to be very similar to that in the prediction space.

The dimensionality of this representation is reduced using the principal component analysis (PCA). The size of the context space after the dimensionality reduction is a tradeoff between the loss of information and the computational demands of a K-means clustering. The value of 10 is practically reasonable. Such low dimensional context space makes the subsequent K-means clustering computationally cheap.

The K-means clustering procedure splits a word class only if the number of words in this class is above an empirically determined threshold $s = 1000$. Each class containing more than s words is thus divided in $K = \lfloor\sqrt{s} + 1\rfloor$ subclasses. The value of this threshold allows us to tune the depth of the clustering tree: small values result in a deep hierarchical structure while a very large value generates a flat tree. The K-means algorithm starts with $4k$ top classes. Each subclass is recursively divided. However, in practice, the one vector initialization used at Step 1 implies that very rare words are represented by very similar or identical feature vectors. In that particular case, the rare words are then naturally grouped in the same class by the first K-means step, and a recursive subdivision is therefore not well suited. The class for rare words is therefore randomly divided.

The method above is not the only way to induce the output word structure, and many other word clustering algorithms have been proposed in the literature. For instance, one can use directly output word classes from Brown's algorithm described in Section 1.3.2. Another possibility is to apply the factorization of the output layer as has been done for RNNLMs (Mikolov et al., 2011c). This approach, referred to as Unigram, is simpler than ours, since it is based on the unigram distribution that does not reveal any relationships between words.

For the experiments in this section, we rather follow the enhanced SOUL training scheme with 4 steps (Algorithm 3) than the original one (Algorithm 2) used in Section 2.4.1.1. The former adds an additional step to better deal with less frequent words. So, after carrying out Step 1 and Step 2, an NNLM that takes the OOS words as output is trained until convergence at Step 3. Finally, at Step 4, a full-vocabulary NNLM is trained. This model has the hierarchical structure at the output part and its parameters are initialized by the parameters of models obtained at the previous steps.

Using Brown and Unigram clusterings, models can be directly trained with Step 4. This scenario is referred to as *single-step* in Table 3.1, where the results obtained with different word clustering schemes are reported. The original enhanced SOUL clustering procedure with all the training steps is referred to as *SOUL*. In this SOUL scenario, neural networks based on Brown and Unigram clusterings also benefit from the information obtained during Step 1 and Step 3 (e.g., using more data to estimate probabilities of OOS words). The original clustering method based on word similarity in the continuous space in the neural network is referred to as NNLM; its perplexities are reported in the last row. It should be noted that both Brown and Unigram approaches provide hierarchical structures, just as the original NNLM method.

Several conclusions can be drawn from the results in Table 3.1. First, models

Table 3.1: *Stand-alone SOUL NNLM perplexity with different word clusterings on the Mandarin dev09_M set.*

clustering type	**4-gram**		**6-gram**	
	single-step	*SOUL*	*single-step*	*SOUL*
Unigram	259	248	229	222
Brown	253	245	225	218
NNLM	-	245	-	220

with NNLM clustering slightly outperform Brown and Unigram clusterings if the latter are computed in the single-step setting. However, the perplexity results in *single-step* and *SOUL* columns should not be directly compared, as all the NNLMs in the latter scenario benefit from the pre-training of neural network parameters in Step 1 and Step 3, as it was mentioned above. Results for the SOUL scenario suggest that taking advantage of the SOUL training approach brings additional improvements for the models based on the Brown and Unigram clustering methods. At the same time, there is no significant difference in perplexity between the three methods in the SOUL scenario. This implies that the way less frequent words are assigned to classes is not very important when the complete SOUL NNLM training is performed.

3.2 Tree clustering configuration

In this section, we investigate the impact of the clustering algorithm: the size of the shortlist, the number of top classes, and the depth of the tree. In these experiments, one model with 300 nodes in the projection layer for each history word and 500 nodes in the hidden layer is trained for each configuration. The model is trained according to the enhanced SOUL training scheme. At Step 3, we train an NNLM with OOS words until convergence (after 5 epochs). Except for the configuration parameters of tree structures and the hyper-parameters mentioned above, the others are kept as presented in Table 2.1.

Results with NNLMs trained using different sizes for the shortlist are presented in Table 3.2. The *shortlist* column corresponds to different sizes of the shortlist, *top classes* reports the number of top classes of the main softmax layer, *depth* is the depth of the SOUL clustering tree, *alone* is used to refer to the cases when only NNLMs are used and *int.* when NNLMs are interpolated with the baseline n-gram model.

One conclusion that can be drawn by looking at Table 3.2 is that the flat full-vocabulary NNLM, while being computationally very expensive, does similarly or slightly worse than the others. The reason is that, to predict the less frequent words, the shortlist NNLMs back-off to normalized Kneser-Ney estimates, whereas the SOUL NNLMs benefit from the generalization of the clustering tree. The SOUL NNLMs also deliver top results with a relatively small shortlist ($8k$). This may be important in order to save computation time

Table 3.2: *Perplexity for LMs with different shortlist sizes on the Mandarin dev09_M set.*

model			**ppl**		**training time (days)**
shortlist	*top classes*	*depth*	*alone*	*int.*	
4-gram Kneser-Ney					
-	-	-	211		-
6-gram shortlist NNLMs					
8k	-	-	222	178	2.4
12k	-	-	222	175	3.5
25k	-	-	223	171	7.1
6-gram SOUL NNLMs					
8k	4k	3	220	169	3.1
12k	4k	3	219	168	5.6
25k	4k	3	219	168	7.0
flat full-vocabulary 6-gram NNLMs					
all (56k)	0	1	226	171	14.7

and resources, as it is not necessary to train SOUL NNLMs with large shortlists. For example, running similar experiments with shortlists equal or close to the vocabulary size is hardly feasible on larger vocabulary setups[1], e.g., Arabic with 300k vocabulary entries, because of prohibitive training costs.

Table 3.3 reports the perplexity for Mandarin 6-gram SOUL NNLMs with different numbers of top classes in the main softmax layer, SOUL NNLMs with clustering trees of different depths and SOUL NNLMs without shortlist where all words are clustered.

From the first five rows, it can be seen that the number of top classes (from 128 to 4k) does not have much influence on the final perplexity. The results from the next three rows imply the same conclusion for the depth of the clustering tree. On the one hand, training with a flat tree is slow and it yields a higher perplexity. On the other hand, there is small difference between clustering trees of depth two and three. This can be explained by the fact that clustering mostly concern rare words; there is thus little point to perform a deep and fine-grained clustering. However, deeper clustering trees are expected to provide training speed-ups for larger vocabulary tasks as it results in more numerous but smaller softmax layers[2]. The benefit of using the shortlist part in the SOUL architecture is also confirmed as the perplexities reported in the last two rows are larger than the others. It suggests that in the SOUL architecture, frequent words should be treated differently than rarer words.

[1]56k Mandarin vocabulary size can be considered as very moderate

[2]A similar experiment on Arabic shows that a three-level tree indeed provides gains in training time as opposed to the two-level one (5.8 days vs. 6.2 days).

Table 3.3: *Perplexity for 6-gram SOUL NNLMs with different number of top classes and clustering tree depths on the Mandarin dev09_M set.*

model			**ppl**		**training time**
shortlist	top classes	depth	alone	int.	(days)
SOUL NNLMs with different numbers of top classes					
$8k$	128	3	221	169	2.4
$8k$	256	3	218	168	3.5
$8k$	$1k$	3	220	169	3.6
$8k$	$2k$	3	220	169	2.7
$8k$	$4k$	3	220	169	3.1
SOUL NNLMs with different clustering tree depths					
all	0	1	226	171	14.7
$8k$	$4k$	2	220	169	3.0
$8k$	$4k$	3	220	169	3.1
SOUL NNLMs without shortlist part					
0	$4k$	2	242	176	2.2
0	$4k$	3	240	176	2.4

3.3 Towards deeper structure

In previous experiments on the Mandarin ASR task, though SOUL NNLMs were shown to be able to handle vocabularies of arbitrary sizes and to outperform shortlist based NNLMs, from the computational point of view, they are as time consuming as the latter. This issue hinders the use of neural network with deeper structures. Therefore, in this section, we first estimate the complexity of SOUL NNLMs in order to show that the size of the shortlist and the number of top classes can be tuned to significantly reduce the computational time. The shallow structure issue is then discussed. As deeper NNLMs can be trained in a reasonable time with the new configuration, experiments are carried out in order to investigate SOUL NNLMs with deep structures. Some conclusions are finally drawn from the experimental results.

Complexity issue In the experiments presented so far, SOUL NNLMs use a shortlist of $8k$ words and $4k$ top classes for OOS words, which makes a total of $12k$ units in the main softmax layer. This choice was mainly meant to allow for a fair comparison with shortlist NNLMs. Because the time complexity is dominated by the main softmax layer size, the computational cost is similar to a $12k$ shortlist NNLM. This cost can be roughly estimated by counting F, the number of floating point operations (Flops) needed, as indicated in Equation (1.99), where V^o now denotes the size of the main softmax layer:

$$F = ((n-1) \times M + 1) \times H + (H+1) \times V^o$$

The most important factors are $(n-1) \times M \times H$ and $H \times V^o$. Assuming that $V^o \gg (n-1)M$, the cost can be considered to be linear in V^o. However, this assumption does not really stand in our experiments due to the use of large values for M (500) and n (6). The value of V^o has nevertheless an important influence on the complexity. Consequently, a non-trivial tradeoff must be found for a shortlist NNLM: V^o needs to be limited for tractability purpose, and, at the same time, V^o must be large enough to ensure a vocabulary size that implies a sufficient n-gram coverage. On the contrary, the SOUL NNLM can predict all possible n-grams and the shortlist is used at the first training step mainly to bootstrap the model and to build the clustering tree for the output structure. As a result, there is no evidence that such a large main softmax layer is necessary for the SOUL NNLM and its size should be tuned as a tradeoff between training time and recognition results.

Shallow structure issue concerns the number of hidden layers. With the hierarchical output structure of the SOUL NNLM, the hidden layer is a complex and highly-varying function of the input layer as it provides input not only for one but for thousands of softmax layers. For example, each Mandarin SOUL NNLM with a shortlist of 8k words and 4k top classes setup has three thousands output softmax layers. To enforce the model capacity, we can either increase the size of the hidden layer or modify the connections between the hidden and the input layers. The latter alternative seems preferable as using a large hidden layer drastically increases the number of parameter and slows down the system. Deep neural networks with several hidden layers usually need to be pre-trained in an unsupervised or a semi-supervised way following a rather complex procedure (Bengio, 2009) to avoid the convergence to poor local extrema, leading to a weak generalization capacity. As adapting the same techniques for NNLMs is not straightforward, in the present study, the pre-training steps are ignored. For that reason, it can be considered as a first step towards tuning language modeling into deep learning. Several architectures of the SOUL NNLM are considered: only one hidden layer; two hidden layers with a large first layer and three hidden layers.

Experimental results Again, the GALE Mandarin ASR setting is used to evaluate the performance of different models. Following the enhanced SOUL training scheme, each NNLM is trained on average with 25M example/epoch after resampling the training data. For each test configuration, three 6-gram NNLMs are trained and interpolated. The order of NNLMs is 6 as using longer context with NNLMs has been shown to be useful for this data. Each of these three NNLMs differs in training data resampling. For all experiments, the nonlinear activation of hidden layers is sigmoid, the projection dimension is set to 500. All parameters are summarized in Table 3.4.

Experiments are first performed in order to examine the size of the output structure. Two configurations are considered. The first one, which was used in our previous work has a main softmax layer of 12k units (8k as the number

parameter	**SOUL NNLM**
training data size	$\approx 3200M$
vocabulary size	56,052
number of models	3
order (n)	6
projection space (M)	500
nonlinear activation	sigmoid
number of examples/epoch	$\approx 25M$
number of epochs (shortlist)	3
number of epochs (OOS)	5
number of epochs (all words)	10
learning rate	1×10^{-2}
learning rate decay	not used
weight decay	3×10^{-5}
block size	32

Table 3.4: *SOUL NNLM configurations.*

of words in the shortlist and $4k$ as the number of top classes for OOS words). The second configuration is much smaller with $4k$ units in the main softmax layer ($2k$ for the shortlist and $2k$ for the OOS words). These configurations are compared for two SOUL models: the model with one hidden layer consisting of 500 nodes and another one of two hidden layers consisting of 1000 and 500 nodes.

The perplexity and CER results are presented in Table 3.5. Training times are reported in the third column and are computed as an average number of hours needed to train three NNLMs following the enhanced SOUL training scheme. Perplexity results are reported for *dev09* and CER results are on *dev09s* and *eval09*. The first row corresponds to the results obtained with the baseline 4-gram model. The other rows report on the results obtained when rescoring of lattices generated with the baseline system with the 6-gram SOUL NNLMs differing in main softmax sizes and hidden structures as shown in the first and the second columns.

It can be seen in Table 3.5 that the results obtained with one and two hidden layer NNLMs with different sizes of the main softmax layer ($12k$ vs. $4k$) do not expose clear and systematic tendencies. The small differences suggest that the main softmax layer size can actually be reduced at least by a factor of 3 without any influence on the performance. In this way, the computational time can be reduced efficiently since the models with the larger output are approximately two times slower. These results also confirm that the choice of the size for the output structure has a little influence on the final results.

Another peculiarity of the SOUL NNLM that is investigated in this study is the structure of the hidden part of NNLMs. NNLMs with different numbers or different sizes of hidden layers are trained using the same configuration of the main softmax layer ($2k - 2k$). In all cases, the size of the last hidden layer

model		**training time (h)**	**ppl**	**CER**	
main softmax layer	*hidden layers*		*dev09*	*dev09s*	*eval09*
4-gram back-off					
-	-	-	211	9.8	8.9
Adding 6-gram SOUL NNLM					
12k	500	120	159	9.0	8.2
4k	500	55	160	9.1	8.3
12k	$1000-500$	145	152	9.0	8.2
4k	$1000-500$	80	152	**9.0**	**8.1**

Table 3.5: *Perplexity and CER (in %) for SOUL NNLMs with different numbers of hidden layers and different main softmax layer sizes.*

is fixed to 500. Table 3.6 reports the final results with the same notation as in Table 3.5. In all cases, SOUL NNLMs are interpolated with the baseline 4-gram back-off LM.

Adding another hidden layer of 1000 nodes is useful as the NNLMs with two hidden layers of $1000-500$ achieves 5% of perplexity reduction over the models with only one hidden layer (the first and the second rows). It is worth noticing that, as shown in Table 3.5, the same improvement of perplexity is observed with the larger main softmax layer. Unfortunately, the perplexity improvements do not translate into consistent CER improvements (while 0.2% of absolute CER reduction is achieved with the main softmax layer of 4k nodes, no gain of CER is found with the larger one). The output structure using the smaller main softmax layer is more compact, because words are clustered in a much smaller number of classes. So the CER results seem suggest that models with more complex output can benefit more from the deep structure.

Moreover, adding one additional hidden layer of 1000 nodes or increasing twice the first hidden layer size, from 1000 to 2000 (the third and the fourth rows) yields no improvement. In terms of training time, the former method is faster (100 compared to 120 hours). These results confirm the difficulty of training a neural network of more than two hidden layers without pre-training. In this case, there is no improvement from using more than one additional hidden layer of 1000 nodes.

hidden layers	**training time (h)**	**ppl**	**CER**	
		dev09	*dev09s*	*eval09*
500	55	160	9.1	8.3
$1000-500$	80	152	9.0	8.1
$2000-500$	120	150	9.0	8.1
$1000-1000-500$	100	151	9.0	8.1

Table 3.6: *Perplexity and CER (in %) for SOUL NNLM with deep hidden structures.*

3.4 Summary

In this chapter, a GALE Mandarin ASR framework is chosen to carry out experiments. A well-tuned LIMSI Mandarin ASR system is served as a baseline. In this system, a robust 4-gram Kneser-Ney discounted LM with a vocabulary of 56*k* words is trained on about 3.2 billion corpora (without pruning nor cut-offs). In the first section, the experimental results show that, in terms of perplexity, when the complete SOUL NNLM training is performed, there is no significant difference between three word clustering methods, namely NNLM, Brown and Unigam. However, the SOUL NNLM training regime is important as it is shown to bring additional improvements with all word clustering methods.

In Section 3.2, investigation of SOUL NNLM configurations on this setting leads to several conclusions about the characteristics of the SOUL architecture. First, frequent words should be treated separately from the rarer words, even though the size of the shortlist is better kept small as there is no significant difference in terms of perplexity between the shortlists of 8*k*, 12*k*, 25*k* or 56*k* words. Second, with the shortlist of 8*k*, the number of top classes for OOS words (from 128 to 4*k*) and the depth of the clustering tree (from 1 to 3 levels) do not have much influence on perplexity. The use of clustering tree itself is important since it provides faster training and better perplexities as compared to flat NNLMs.

In our previous work, significant gains were reported with SOUL NNLMs over shortlist based NNLMs for this GALE Mandarin ASR task. However, the computational costs were the same due to the similar size of the main softmax layer of both types. Several refinements of the structure of SOUL NNLMs are therefore proposed. Following the recently proposed enhanced SOUL training scheme, it is shown that by reducing the size of the output structure (from 8*k* to 2*k* words in the shortlist and from 4*k* to 2*k* top classes for the OOS words), the inference and training times can be drastically reduced without any impact on recognition performance. The resulting SOUL NNLMs achieve better performances and are two times faster than the shortlist based ones.

The limits of shallow neural network architectures are finally investigated. The experimental results indicate that, using one large additional hidden layer leads to 5% reduction in perplexity. On the contrary, using more hidden layers does not bring additional improvements. It may point to the fact that the lack of pre-training prevents deep models from escaping poor local extrema. It thus looks promising to investigate advanced techniques for deep learning similar to the ones presented in (Hinton, Osindero, and Teh, 2006; Bengio, 2009) as a future work. Our study here can therefore be considered as one step towards the deep learning architecture in language modeling.

Chapter 4

Inside the Word Space

Contents

The representation of words in a continuous space allows NNLMs to take into account the similarity between words. This characteristic is often claimed to be the main advantage of NNLMs, as compared to standard back-off LMs. However, only a few analyses on the impact of word spaces on the system performance are available in the literature. In (Collobert and Weston, 2008), useful information encoded in the word space of ranking NNLMs (presented in Section 1.4.1.5) was reported. However, the opposite conclusion is often drawn for standard, feed-forward NNLMs. In the same article, the authors suggested that, compared to standard NNLMs, by using the complete context of a word (before and after) to predict its relevance, ranking LMs can learn much better word spaces. Another example can be taken from (Emami, 2006) where the author found a similar fact by claiming that *"It also seems that the feature vectors obtained after training are only suited for use with the corresponding neural net"* (page 50). As such, it raises a crucial question: *"If NNLMs are not able to learn word spaces, at least as we intuitively want, where do their improvements come from?"*.

To address this issue, Section 4.1 of this chapter is hence devoted to the empirical study on the impact of word spaces on system performances. This study is based on different ways of viewing the information in word spaces induced from two neural network structures for statistical language modeling: the standard models of Bengio, Ducharme, and Vincent (2000) and the LBL models of Mnih and Hinton (2007) (described in Section 1.4.1.1 and Section 1.4.1.2 respectively). The main purpose is to better understand the differences between these models, especially in terms of the word representations that they induce. These results highlight the impact of parameter initialization of word spaces. Based on this study, we will then show that by better learning the relationship between words, improvements of both the speed and the prediction capacity of the standard NNLMs can be achieved. First, a method called *re-initialization* is investigated. Basically, it allows the model to escape from the local extremum the standard model converges to. While this method yields a significant improvement, the underlying assumption about the structure of the model does not meet the requirement of large vocabularies. We therefore introduce a different initialization strategy, called *one vector initialization*. Experimental results show that these novel training strategies can give a better sense to word spaces. For reference, most part of this work was published in (Le et al., 2010). In Section 4.2, several word spaces provided by SOUL NNLMs will then be qualitatively analyzed by exploring word neighbors of some random words.

Finally, in Section 4.3, we will briefly analyze the preliminary results for word relatedness measure, another task related to the semantic similarity between words. In this task, the output word embeddings of NNLMs are considered as a useful source of information. By comparing to other state-of-the-art approaches, we will show that SOUL NNLMs can effectively learn some syntactic and semantic information of words.

4.1 Two spaces study

For all the experiments in this section, 4-gram language models are train on a large monolingual corpus containing all the English texts in the parallel data of the Arabic to English NIST 2009 constrained task[1]. It consists of 176 millions of word tokens with 532,557 different word types as the size of the vocabulary. The perplexity is computed with respect to the 2006 NIST test data, which is used here as our validation data. The use of a back-off 4-gram model estimated with the modified Knesser-Ney smoothing on the training data achieves a perplexity of 141 on the validation data.

4.1.1 Convergence study

As shown in Section 1.4.1.2, the main difference between standard NNLMs and LBLs is that, while NNLMs have two different word spaces (context and prediction), in LBLs, they are bound to be the same (see Figure 4.1). As a result, by comparing the two sorts of NNLMs, we can have some intuitive views on the relation between two spaces. In order to do that, we train them in the same setting: we choose to consider a small vocabulary comprising the 10,000 most frequent words. The same vocabulary is used to constrain the words occurring in the history and the words to be predicted, meanings that the input and output vocabularies are identical. To make a fair comparison, the size of the hidden layer, which is also the dimension of the prediction space, and the dimension of the context space are equally set to 200 ($M = H = 200$).

Figure 4.2 displays the perplexity convergence curve measured on the validation data for the standard and LBL models. We observe that the LBL

[1]http://www.itl.nist.gov/iad/mig/tests/mt/2009

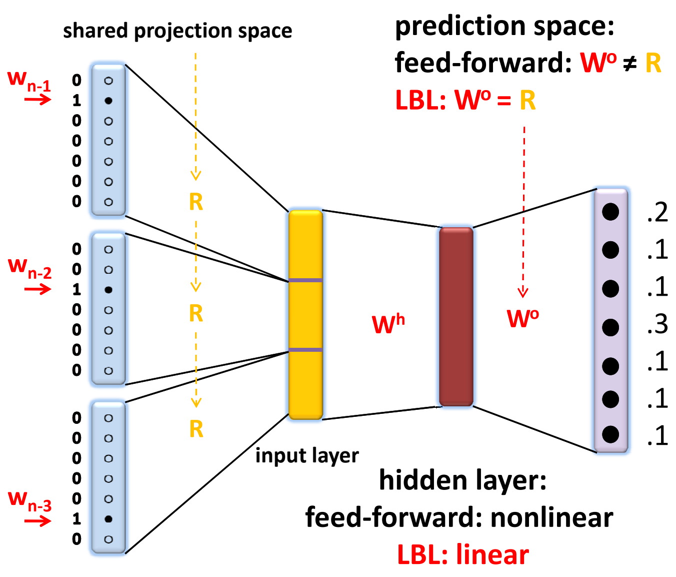


Figure 4.1: *Feed-forward NNLM and LBL architectures.*

model converges faster than the standard model: the latter needs 13 epochs to reach the stopping criteria, while the former only needs 6 epochs. However, upon convergence, the standard model reaches a lower perplexity than the LBL model. The convergence perplexities after combination with the standard back-off model[2] are also provided in Table 4.2 (in the first two rows).

With a smaller number of parameters, the LBL model cannot capture as many characteristics of the data as the standard model, but it converges faster. This difference in convergence can be explained by the scarcity of the updates made in the projection matrix $\mathbf{R}$ in the standard model: during back-propagation, only those weights that are associated with words in the history are updated. By contrast, each training sample updates all the weights in the prediction matrix $\mathbf{W^h}$.

4.1.2 Word representation analysis

To deepen our understanding of these distributed representations, we propose to further analyze the induced word embeddings by examining, for some randomly selected words, the five nearest neighbors (according to the Euclidean distance) in the context space and in the prediction space of the two models. Results are presented in Table 4.1.

[2]as done with shortlist NNLMs, described by Equation (1.100) in Section 1.4.3.2.

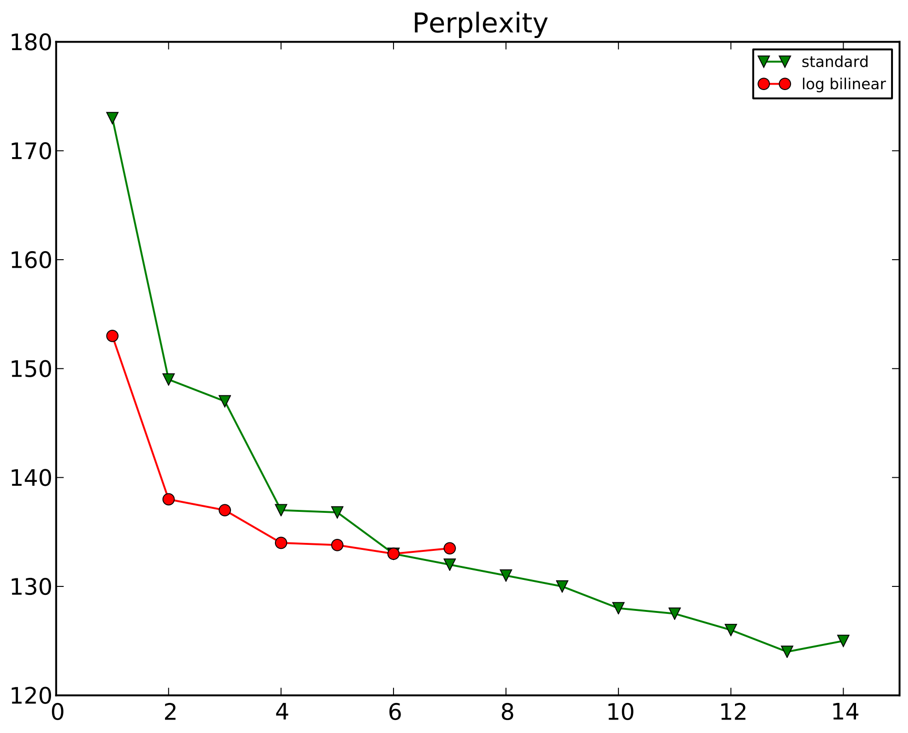


Figure 4.2: *Convergence rate of the standard and the LBL models evaluated by the evolution of the perplexity on a validation set.*

If we first look at the standard model, the global picture is that, for frequent words (*is*, *are*, and, to a lesser extent, *have*), both spaces seem to define meaningful neighborhoods, corresponding to semantic and syntactic similarities; this is less true for rarer words, where we see a greater discrepancy between the context and prediction spaces. For instance, the date *1947* seems to be randomly associated in the context space, while the 5 nearest words in the prediction space form a consistent set of dates. The same trend is also observed for the proper name *Castro*. Our interpretation is that, for less frequent words, the projection vectors are hardly ever updated and remain close to their original random initialization, thus hindering the emergence of meaningful clusters.

By contrast, the similarities in the (unique) projection space of the LBL remain consistent for all frequency ranges, and are very similar to the prediction space of the standard model. This seems to validate our hypothesis that in the standard model, the prediction space is learned much faster than the context space and corroborates our interpretation of the impact of the scarce updates of rare words. Another possible explanation is that there is only rather indirect relation between the context space and the objective function: the context space is learned only indirectly by back-propagation. As a result, due to the random initialization of the parameters and to the sparsity of counts, many vectors in **R** might be blocked in some local maxima, meaning that similar vectors cannot be grouped in any consistent way and that the induced similarity is more “loose”.

Table 4.1: *The 5 nearest neighbors in the word spaces of the standard and LBL language models.*

word/ frequency	model	space	5 nearest neighbors
is	standard	context	was are were be been
900,350	standard	prediction	was has would had will
	LBL	both	was reveals proves are ON
are	standard	context	were is was be been
478,440	standard	prediction	were could will have can
	LBL	both	were is was FOR ON
have	standard	context	had has of also the
465,417	standard	prediction	are were provide remain will
	LBL	both	had has Have were embrace
meeting	standard	context	meetings conference them 10 talks
150,317	standard	prediction	undertaking seminar meetings gathering project
	LBL	both	meetings summit gathering festival hearing
Imam	standard	context	PCN rebellion 116. Cuba 49
787	standard	prediction	Castro Sen Nacional Al- Ross
	LBL	both	Salah Khaled Al- Muhammad Khalid
1947	standard	context	36 Mercosur definite 2002-2003 era
774	standard	prediction	1965 1945 1968 1964 1975
	LBL	both	1965 1976 1964 1968 1975
Castro	standard	context	exclusively 12. Boucher Zeng Kelly
768	standard	prediction	Singh Clark da Obasanjo Ross
	LBL	both	Clark Singh Sabri Rafsanjani Sen

4.1.3 Learning techniques

In the previous section, we observed that slightly better results can be obtained with the standard rather than with the LBL model. The latter is however much faster to train and seems to induce better projection matrices. Both effects can be attributed to the particular parameterization of this model, which uses the same projection matrix both for the context and for the prediction spaces. In this section, we propose several new learning regimes that allow us to improve the standard model in terms of both speed and prediction capacity. All these improvements rely on the idea of sharing word representations. While this idea is not new (see for instance (Collobert and Weston, 2008)), our analysis enables to better understand their impact on the convergence rate.

4.1.3.1 Re-initialization

The experiments reported in the previous section suggest that it is possible to improve the performance of the standard model by building a better context space. We therefore introduce a new learning regime, called *re-initialization* which aims to improve the context space by plugging in the information on word neighborhoods that emerges in the prediction space. One possible implementation of this idea is as follows:

Algorithm 4 Re-initialization

```
Train a standard model until convergence
Use the prediction space of this model to initialize the context space of a new model;
the prediction space is chosen randomly
Train this new model
```

The evolution of the perplexity with respect to training epochs for this new method is plotted on Figure 4.3, where we only represent the evolution of the perplexity during the third training step. As can be seen, at convergence, the perplexity of the new model is about 10% smaller than the perplexity of the standard model.

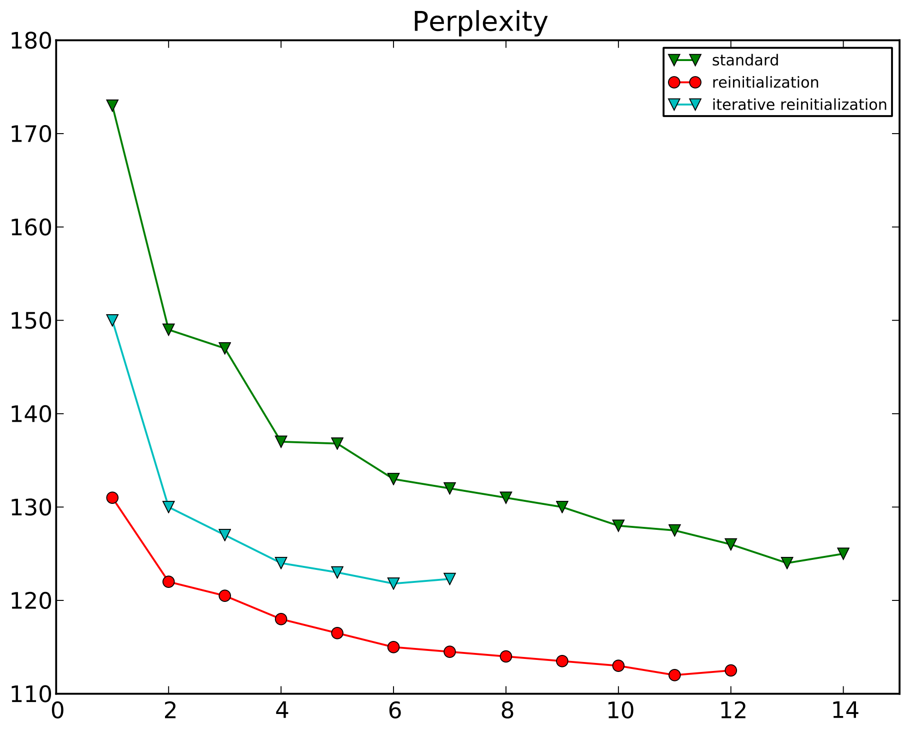


Figure 4.3: *Evolution of the perplexity on a validation set for various initialization regimes.*

This result can be explained by considering the re-initialization as a form of annealing technique: re-initializing the context space allows escaping from the local extrema the standard model converges to. The fact that the prediction space provides a good initialization of the context space also confirms our analysis that one difficulty with the standard model is the estimation of the context space parameters.

4.1.3.2 Iterative re-initialization

The re-initialization policy introduced in the previous section significantly reduces the perplexity, at the expense of a longer training time, as it requires to successively train two models. As we now know that, the parameters of the prediction space are faster to converge, we introduce a second training regime called *iterative re-initialization* which aims to take advantage of this property. We summarize this new training regime as follows:

Algorithm 5 Iterative re-initialization

```
repeat
  Train the model for one epoch
  Use the prediction space parameters to reinitialize the context space
until Convergence
```

This regime yields a model that is somewhat in-between the standard and LBL models as it adds a relationship between the two representation spaces, which is lacking in the former model. This relationship is however not expressed through the tying of the corresponding parameters; instead we let the prediction space guide the convergence of the context space. As a consequence, we hope that it can converge as quickly as the one of the LBL model without degrading its prediction capacity.

The results plotted on Figure 4.3 show that this indeed the case: using this training regime, we obtained a perplexity similar to the one of the standard model, while at the same time reducing the total training time by more than a half, which is of great practical interest[3].

Figure 4.4 displays the perplexity convergence curve measured on the training data for the standard learning regime as well as for the re-initialization and iterative re-initialization. These results show the same trend as for the perplexity measured on the validation data, and suggest a regularization effect of the re-initialization schemes rather than allowing the models to escape local optima.

4.1.3.3 One vector initialization

The new training regimes introduced above outperform the standard training regime both in terms of perplexity and of training time. However, exchanging

[3]Each epoch lasts approximately 8 hours on a 3GHz Xeon processor.

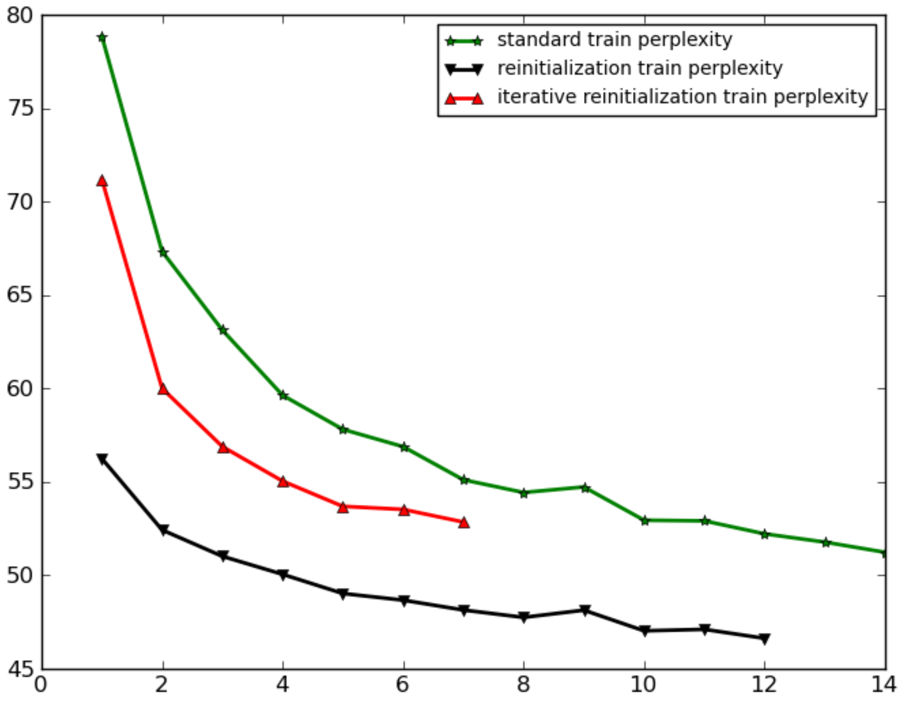


Figure 4.4: *Evolution of the perplexity on the training data for various initialization regimes.*

information between the context and prediction spaces is only possible when the same vocabulary is used in both spaces. This configuration of standard NNLMs[4] is not realistic for very large-scale tasks because increasing the number of predicted word types is much more computationally demanding than increasing the number of types in the context vocabulary. Thus, the practical requirement to use shortlists, as indicated in Section 1.4.3.2 leads to the fact that, the former vocabulary is typically order of magnitudes larger than the latter, which means that the re-initialization strategies can no longer be directly used.

It is nonetheless possible to continue drawing some inspiration from the observations made in Section 4.1.2, and, crucially, to question the random initialization strategy. As discussed above, this strategy may explain why the neighborhoods in the induced context space for the less frequent types are difficult to interpret. As a straightforward alternative, we consider a different initialization strategy where all the words in the context vocabulary are initially projected onto the same (random) point in the context space. The intuition

[4]They are NNLMs without SOUL structure at the output. Here, we do not consider SOUL NNLMs because they have only one word space: a context one (see Chapter 2 for more details).

is that, it will be easier to build meaningful neighborhoods, especially for rare types, if all words are initially considered similar and only diverge if there is sufficient evidence in the training data to suggest that they should be distinguished. This model is termed the *one vector initialization* model.

To validate this approach, we compare the convergence of a standard model trained (with the standard learning regime) with the one vector initialization regime. The context vocabulary is defined by the 532,557 words occurring in the training data and the prediction vocabulary by the 10,000 most frequent words. The other parameters are the same as in the previous experiments. Based on the curves displayed on Figure 4.5, we can observe that the model obtained with the one vector initialization regime outperforms the model trained with a random initialization. Moreover, the latter reaches convergence in only 14 epochs, while the learning regime we propose only needs 9 epochs.

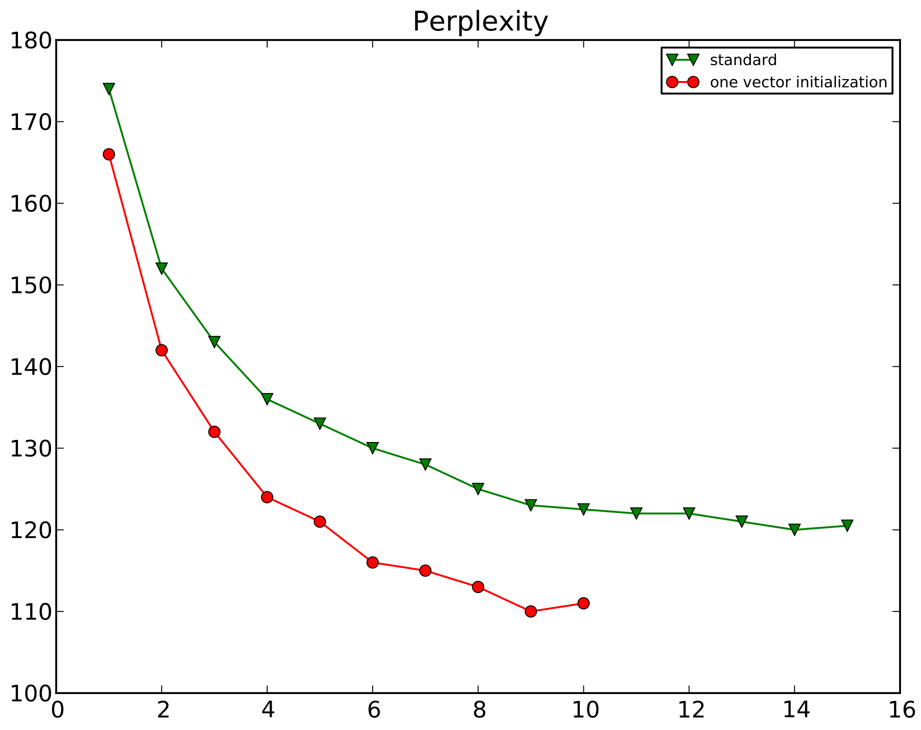


Figure 4.5: *Evolution of the perplexity on a validation set for all-*10,000 *standard and one vector initialization models.*

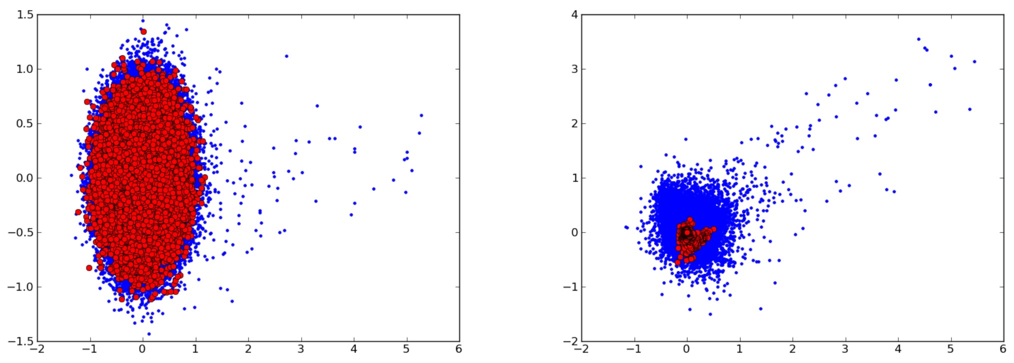
To illustrate the impact of our initialization scheme, we also use a principal component analysis to represent the induced word representations in a two dimensional space. Figure 4.6 represents the vectors associated with numbers[5]

[5]are all the words consisting only of digits, with an optional sign, point or comma such as: *1947*; *0,001*; *-8,2*.

Table 4.2: *Summary of the perplexity (ppl) results measured on the same validation set with the different continuous space language models. For all of them, the probabilities are combined with a back-off n-gram model.*

$\mathcal{V}_c$ **size**	**model**	**# epochs**	**ppl**
10000	log bilinear	6	239
	standard	13	227
	iterative reinitialization	6	223
	reinitialization	11	211
all	standard	14	276
	one vector initialization	9	260

in red, while all the other words are represented in blue. Two different models are used: the standard model on the left, and the one vector initialization model on the right. We can observe that, for the standard model, most red points are scattered all over a large portion of the representation space. On the opposite, for the one vector initialization model, points associated with numbers are much more concentrated: this is simply because all the points are originally identical, and the training aims to spread the point around this starting point. We also create the nearest neighbor lists reported in Table 4.3, in a manner similar to Table 4.1. Clearly, the new method seems to yield more meaningful neighborhoods in the context space.



(a) *with the standard model* **(b)** *with the one vector initialization model*

Figure 4.6: *Comparison of the word embeddings in the context space for numbers (red points).*

4.2 Word representation analysis for SOUL NNLMs

As done in Section 4.1.2, the similarity between words in the projection space of SOUL NNLMs can also be analyzed by finding the nearest neighbors of words

Table 4.3: *The 5 nearest neighbors in the context space of the standard and one vector initialization language models.*

word/ frequency	**model**	**5 nearest neighbors**
is	standard	was are were been remains
900,350	one vector init.	was are be were been
conducted	standard	undertaken launched $270,900 Mufamadi 6.44-km-long
18,388	one vector init.	pursued conducts commissioned initiated executed
Cambodian	standard	Shyorongi $3,192,700 Zairian depreciations teachers'
2381	one vector init.	Danish Latvian Estonian Belarussian Bangladeshi
automatically	standard	MSSD Sarvodaya $676,603,059 Kissana 2,652,627
1528	one vector init.	routinely occasionally invariably inadvertently seldom
Tosevski	standard	$12.3 Action,3 Kassouma 3536 Applique
34	one vector init.	Shafei Garvalov Dostiev Bourloyannis-Vrailas Grandi
October-12	standard	39,572 anti-Hutu $12,852,200 non-contracting Party's
8	one vector init.	March-26 April-11 October-1 June-30 August4
3727th	standard	Raqu Tatsei Ayatallah Mesyats Langlois
1	one vector init.	4160th 3651st 3487th 3378th 3558th

according to the Euclidean distance. In this section, we pick one of SOUL NNLMs developed for an Arabic ASR task which was presented in Chapter 2, as well as a model for English which uses data from the WMT evaluation campaign 2012 to evaluate resulting word spaces. French, German or Spanish native speakers are invited to go to Appendix B to have the similar analyses on their-own native languages.

For Arabic, Table 4.4 includes some words with close concepts or sharing similar functions, that are close in the SOUL projection space. Words in the first column are the words for which "similar" words are determined (labeled *nearest neighbors*). The nearest neighbors are presented first with more distant words near the end of the lists. The Arabic words are MADA-decomposed units represented with a slightly modified MADA (Roth et al., 2008) notation used at LIMSI along with English glosses. The indices *(m)* and *(f)* stand for the grammatical markers for gender (masculine/feminine), an information that is encoded in Arabic words. For English, Table 4.7 is built in the similar way, with random words selected among the 10,000[6] most frequent words.

[6]This number is chosen just to guarantee that the created list will be not filled with rare words due to the uniform selection, which makes the analysis more difficult.

Table 4.4: *Examples of nearest MADA-decomposed Arabic words according to the SOUL NNLM projection space.*

word	nearest neighbors
Aldwlp	AlHwmp - bnwknA - mWsstnA - jAmctkm - mSArfhA - bldytnA - nqAbth - vwrtnA - cAlqDyp - cAlclAqAt - mSArfnA - dwltkm - Aldwylp - wzArthA - mHAkmnA
country	district - our banks - our institutions - your universities - its banks(f) - our town - syndicate - our war - about affair - about relations - our banks - your country - small country - its ministry(f) - our tribunals
AlmADy	AlmADyp - AlHAly - AlmADyyn - AlmnSrmyn - mggAk - mWxrA - ywmgAk - jwkAnwfytX - AlmADyn - mZAhry - EwAxrh - mwskwfytX - AlfAetyn - cAmgAk - mgAk
past(m)	past(f) - present - previous(pl) - previous(pl) - since - late - that day - Dzukanovitch - previous(pl) - demonstrators - last - Mouskovitch - previous(pl) - that year - since
jAC	yjyC - EtY - yEty - wAtY - mAjAC - mAwrd - yEtyAn - jACA - tEtY - jAChA - jACtA - Astwqfny - AstcjlwA - ycksh - wkAn - yje - ytbdY
arrived	will arrive - come - he will come - and come - he didn't come - destination - they will come - they arrived - she will come - smb/smth(m) came/happened to her - smb/smth(f) came/happened to her - I stopped (because of smth.) - they hurried up - he met - he was - he will arrive - it/he will start
Intrnt	mwdm - blwtwv - HAswbyp - myJAhyrtz - lAbtwb - AlwAb - kwmywnykyXnz - HwAsb - brmjyp - Abswn - nAfyJytr - mdmjA - AllAbtwb - HAswby - mdmj - AlHAswbyp - byksl
Internet	modem - bluetooth - computer - megahertz - laptop - web - communications - computers - programming - Epson - navigator - compact(f) - laptop - my - computer - compact(m) - computer - pixel

In many cases, we observe that words with semantic or grammatical similarities also have similar representations in the projection space. Thus, neural networks following SOUL training scheme seem to capture some of the similarities that exist between words.

4.3 Word relatedness task

There exist other tasks that can be used to evaluate the quality of word space in more direct fashions. For example, in (Turian, Ratinov, and Bengio, 2010), word spaces obtained using different methods: Brown clustering (Section 1.3.2), HLBL models (Section 1.4.1.3) and ranking NNLMs (Section 1.4.1.5) were compared together in NER and chunking tasks. Here, we use another task aiming at learning word semantic similarity from text data. Given a list of word pairs, one needs to assign to each pair a score that measures the similarity (or more exactly speaking, the relatedness) of its words. This task provides useful information for more sophisticated tasks such as information retrieval, word sense disambiguation, text clustering among others. The performance of a method is evaluated based on the correlation between its provided scores and the available average of human judgments. For example, a couple of words: "love" and "sex" has a score of 6.77/10 according to human judgments in WordSim-353. In the literature, the Spearman's rank order correlation is usually used (the larger is the better). In order to easily compare our method to others, we use the two data sets: WordSim-353 (Finkelstein et al., 2001) and MTurk-287 (Radinsky et al., 2011) on which the results for most state-of-the-art approaches are available.

4.3.1 State-of-the-art approaches

Most current approaches rely upon the use of various linguistic resources: large sets of documents for corpus-based approaches; lexical databases in thesaurus-based approaches. Each word is represented as a vector in a high dimension space. The similarity of each pair of words is then often measured as the cosine similarity of their vectors in the space, though other measures are also available. For example, in corpus-based approaches, Latent Semantic Analysis (LSA) (described at the end of Section 1.3.5) or Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch, 2007) use the context words, documents, topics or concepts of words; Temporal Semantic Analysis (TSA) (Radinsky et al., 2011) is based on the study on patterns of word usage over time. For reference, in (Yih and Qazvinian, 2012), the authors give an outline of the state-of-the-art approaches and propose a method to incorporate some of them to yield better results. In summary, most existing methods use information from more advanced and sophisticated knowledge sources compared to raw text data, which is the only source of information that NNLMs learn from.

Recently, in (Huang et al., 2012), several neural network architectures that can automatically learn word space (see Figure 4.7) have been investigated. The goal is to score the relevance of the word in the context. The first type of model is similar to the ranking NNLMs described in Section 1.4.1.5, but the target word is used instead of the middle word. The training is performed so as to increase the score of a positive n-gram example (obtained from the text) and to decrease the score of the corresponding negative example (obtained by replacing the last word by another word in the vocabulary drawn from the uniform distribution). The model referred to as *ranking LM* is represented in the left part of Figure 4.7; and delivers a local score score_l. The authors also proposed another architecture which has an additional part used to deal with the global context (in the right part of Figure 4.7). This part provides a global score, score_g, which is a function of the word representation and of the representation of its document, computed as the average values of all words occurring in the document. In this new model, the final score is obtained by summing the local and global scores. Moreover, in order to capture homonymy and polysemy, the authors proposed to use multiple vectors to represent a word. Therefore, for each word in the original vocabulary, its occurrences are automatically assigned to one label using the K-means clustering algorithm applied to the vector representation of their contexts[7]. After that, each word occurrence is re-labeled with its cluster tag. Considering each original word and its label as new word type, we use the new corpus to learn multiple representations of the words in the original vocabulary. The similarity score between two words is then computed as the average cosine similarity of all possible pairs of their vector representations. We call *multiple vectors* the model that takes into account the global context information and that makes use of multiple word representations. We use *multiple vectors wo stop words* to refer to the same model when trained without stop words[8]. The proposed models were shown to achieve a comparable result on the WordSim-353 task (see Table 4.5).

Taking directly the final induced word space of NNLMs without any modification neither on their architecture nor on their training fashion, we will show that they can also yield promising results on this particular task.

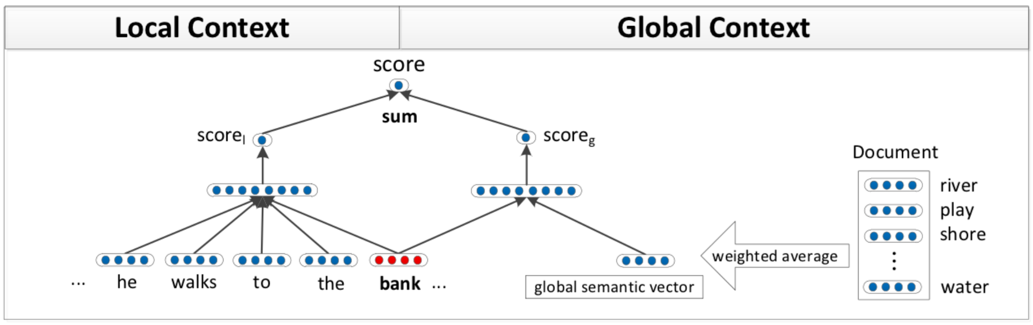


Figure 4.7: *Neural network architecture described in (Huang et al., 2012). The word **bank** is the word that needs to be discriminated using its left context words and its document. This figure is borrowed from (Huang et al., 2012).*

4.3.2 Experimental evaluation

The two data sets WordSim-353 and MTurk-287 are used to evaluate system performances. Training data is the Wikipedia snapshot up to Nov 2010, described in (Shaoul and Westbury, 2010). This corpus is further segmented into sentences, lowercased and then tokenized using the scripts from the Moses machine translation toolkit (Koehn et al., 2007). All sentences that are too short

[7]They are computed as a weighted average of the context words' vectors.
[8]The most frequent words which are considered to be less informative, e.g., is, the, a, an ...)

(containing less than 5 words) or too long (having more than 100 words) are filtered. The resulting data set contains about 50 million sentences and about 1.1 billion words. Two sets of 2500 randomly selected sentences are excluded to form a validation data and a test data. The remaining part is used as training data. The vocabulary contains the 688,017 words which occur more than 10 times in the corpus.

We decide to consider several configurations for SOUL NNLMs. All models are trained following the SOUL enhanced learning algorithm (Algorithm 3 presented in Section 2.3) in two cases: with or without reinitializing the projection space at the beginning of each step. The order of models is 10, the shortlist size end the number of top classes are both set to 2000. We use one hidden layer and fix its size to be equal to the dimension of the word space. The number of epochs for all steps is 2, as using a larger value does not bring significant improvements.

Table 4.5: *Word relatedness evaluation of state-of-the-art approaches on the WordSim-353 and MTurk-287 data sets. Performance is presented as Spearman's rank order correlation coefficient* $\times 100$*. (i) means that this result was taken from (Huang et al., 2012), (ii) from (Gabrilovich and Markovitch, 2007) and (iii) from (Radinsky et al., 2011).*

model	WordSim-353	MTurk-287
LSA (ii)	56	-
ESA (ii)	75	59
TSA (iii)	80	63
ranking LM (i)	55	-
multiple vectors (i)	64	-
multiple vectors wo stop words (i)	71	-

For reference, state-of-the-art results are represented in Table 4.5. Note that the dimension of the word space used by these neural networks is 50. The final results of our NNLMs are shown in Table 4.6. The column **step** denotes which step of Algorithm 3 the model is taken. Recall that models at Step 1 are trained with n-grams that end with in-shortlist words. Models at Step 3 only deal with n-grams ending with OOS words. Models at Step 4 are trained with all possible n-grams as they take the whole vocabulary as output.

The first three rows of Table 4.6 report the experimental results in the case where the same data resampling rate is used at all steps. Moreover, to make a fair comparison, we reinitialize the projection space at the beginning of the training phase for Step 3 and Step 4. These results indicate that the word space obtained from the model of Step 3 is clearly better. This fact suggests that n-grams which end with less frequent words carry a rich information.

With the same dimension of the word space (50), the performance of SOUL NNLMs is similar to the one of ranking NNLMs. Despite the different configurations from different articles that hinders a totally fair comparison, they clarify the assertion, claimed in (Collobert and Weston, 2008), that ranking

Table 4.6: *Word relatedness evaluation of NNLMs with different configurations on the WordSim-353 and MTurk-287 data sets. Performance is presented as Spearman's rank order correlation coefficient* $\times 100$.

dimension	step	resampling rate	activation	WordSim-353	MTurk-287
reinitializing space at each step					
50	1	15%	sigmoid	39	39
50	3	15%	sigmoid	52	53
50	4	15%	sigmoid	48	46
normal training					
50	1	15%	sigmoid	39	39
50	3	15%	sigmoid	52	52
50	4	15%	sigmoid	54	53
more data					
50	3	50%	sigmoid	57	57
larger space					
200	3	50%	sigmoid	62	62
linear activation					
200	3	50%	linear	70	65

NNLMs with their more simple structure and their use of negative examples make the learning of word space easier. In fact, the word embeddings learned from normal NNLMs are not worst at all and they work at least as well as the one induced from ranking LMs for this task.

The three following rows are the results of Step 1, Step 3 and Step 4 using the normal training scheme, meaning that their projection space benefits from the learning of the previous steps. We add the result of Step 1 only for reference as it is the same as in the first row. By comparing rows 3 and 5, we see that the result of Step 4 is significantly improved. On the contrary, the result of Step 3 is almost unchanged (rows 2 and 4). It again demonstrates that the information learned at Step 3 is the most important.

The next row reports the result of Step 3 with more data, using a resampling rate of 50%. With the same rate, we cannot train models of Step 1 and Step 4 due to the memory and time problems. It should be noted that with this configuration, we use a comparable number of training examples after resampling (130 millions/epoch) as for the previous configurations of Step 1 and of Step 4. So with the same training time, the capacity of using much higher resampling rate is an advantage of Step 3. A large gain (5 points) indicates that more data is actually better. At this level and with 2 epochs, resampling is nearly not used, so this result is on top of its performance.

The jump from 57 to 62 when increasing the projection dimension of SOUL NNLM from 50 to 200 demonstrates that for this task, 50 is not sufficient, at least for SOUL NNLMs. On the contrary, other experimental results, which are not reported here, indicate that word spaces with a dimension higher than 200 does not achieve better results.

At the hidden layer, using linear activation is better than using sigmoid activation (the last two rows) in spite of its worse performance on perplexity (3004 compared to 2840[9]) maybe because its linear relation between the target function and the word space favors the use of cosine similarity. It seems to indicate that for more complex word spaces induced by nonlinear models, more complicated similarity measures should be investigated.

As a final conclusion, the best results of SOUL NNLMs (the last row) on the two data sets are very competitive with the best reported results despite the fact that, unlike most other approaches, they only use local information from the raw texts.

4.4 Summary

In this chapter, we first carefully analyze the impact of word spaces induced by standard and LBL models on the performance to demonstrate that, unlike in the original article (Mnih and Hinton, 2007), LBLs do not outperform standard ones. In standard NNLMs, there are two word spaces used to represent the two different roles of a word (context and prediction) plays in language modeling. Moreover, thanks to their similarity, the faster convergence of the prediction space can enforce the convergence of the context space. Three new methods for training NNLMs are proposed and shown to be efficient. This work highlights the impact of the initialization and the training scheme for NNLMs. Both our experimental results and our new training methods are closely related to the pre-training techniques introduced by (Hinton, Osindero, and Teh, 2006).

Following the SOUL approach, the induced word spaces of NNLMs are then proved to essentially encode some loose semantic and syntactic word similarities in a variety of languages. The word embeddings provided by SOUL NNLMs are also interesting for other NLP tasks. The promising results obtained in our attempts at the word relatedness measure task show that SOUL NNLMs perform not only well in its normal task but could also be helpful in other semantic NLP tasks.

[9]They are quite large because at Step 3, the models only predict OOS (less frequent) words.

Table 4.7: *Examples of nearest neighbors according to the SOUL NNLM projection space for English.*

word	**nearest neighbors**
defendant	plaintiff - accuser - co-accused - co-conspirator - perpetrator - petitioner - co-defendant - defendants - interrogator
type	types - kind - kinds - pervasiveness - overabundance - multiplicity - blob - LOTS - troves
adjusted	unadjusted - adjusting - pro-forma - Adjusted - Segment - reconciles - adjusts - LTM - Diluted
Parker	Wilcox - Udrih - Pondexter - Gaither - Dykes - Foye - Delfino - Felton - Humphries
Eds	EDS - EDs - Eds. - FIX - UPDATE - HT - dateline - NOTE - CORRECT
critics	detractors - skeptics - opponents - sceptics - admirers - cynics - backers - proponents - purists
%	percent - thirds - 10pc - fifths - cent - four-fifths - one-fifth - two-fifths - one-third
responsibilities	prerogatives - accountabilities - competencies - competences - endeavors - deliverables - inadequacies - dilemmas - certifications
seminar	symposium - workshop - Seminar - Symposium - webinar - expo - exposition - seminars - treatise
pricing	franchising - cashflow - provisioning - repricing - retailing - customization - sourcing - mis-selling - tracability
Johannesburg	Pretoria - Maputo - Bloemfontein - Durban - Luanda - Bulawayo - Polokwane - Dili - Brasilia
Sen.	Senator - Sens. - Sen - Congressman - senator - Assemblyman - Congresswoman - Rep. - Chairwoman
writing	penning - composing - write - co-writing - authoring - faxing - collating - coauthoring - rereading
deliver	delivering - delivers - delivered - peddle - furnishes - distribute - proffering - impart - hand-delivered
triumph	triumphing - triumphed - rebirth - triumphs - romp - immortality - hegemony - fervour - slip-up
Lions	Springboks - Wallabies - Waratahs - Boks - Cheetahs - Buccaneers - Dragons - Brumbies - Crusaders
criticised	criticized - lambasted - faulted - rebuked - castigated - chastised - criticising - criticise - derided
assurance	reassurances - assurances - reassurance - inference - proviso - certainty - contentions - pretence - affirmation
actively	energetically - diligently - proactively - avidly - constructively - earnestly - intensively - collaboratively - meaningfully
AVENUE	Contemporaries - Appetizers - Cantina - SE1 - $5.20 - Westtown - WC2 - MATEO - Suffern

Chapter 5

Measuring the Influence of Long Range Dependencies

Contents

A difference between conventional and neural network language models that has often been overlooked is the ability of the latter to fare with extended contexts (Schwenk and Koehn, 2008; Emami, Zitouni, and Mangu, 2008); in comparison, standard n-gram LMs rarely use values of n above 4 or 5, mainly because of data sparsity issues and the lack of generalization of the standard maximum likelihood estimates, notwithstanding the complexity of the computations incurred by the smoothing procedures (see however (Brants et al., 2007) for an attempt to build very large models with a simple smoothing scheme).

There have been many attempts to increase the context beyond a couple of history words (see e.g., (Rosenfeld, 2000) for a review), notably by trying to take into account syntactic information that better reflect the "distance" between words (Chelba and Jelinek, 2000; Collins, Roark, and Saraclar, 2005; Schwartz et al., 2011). One such interesting proposal avoids the n-gram assumption by globally estimating the probability of a sentence (Rosenfeld, Chen, and Zhu, 2001). This approach relies on a maximum entropy model which incorporates arbitrary features. No significant improvements were however observed with this model, a fact that can be attributed to two main causes: first, the partition function which acts as a normalizer cannot be computed exactly as it involves a sum over all the possible sentences; second, it seems that data sparsity issues for this model are also adversely affecting the performance.

The recent attempt of Mikolov et al. (2011c) to resuscitate recurrent neural network architectures goes one step further in that direction, as a recurrent network simulates an unbounded history size, whereby the memory of all the previous words accumulates in the form of activation patterns on the hidden layer. RNNLMs (described in detail in Section 1.4.1.4) have been proved to outperform feed-forward NNLMs on small and medium tasks. However, this approach cannot handle large training corpora as easily as n-gram models, which makes it difficult to perform a fair comparison between these two architectures and to assess the real benefits of using very long contexts. To make the comparison feasible, we will try to adapt efficient training techniques developed for n-gram NNLMs, notably SOUL NNLMs, to the case of RNNLMs.

This chapter contains two main contributions. We first analyze the architecture of NNLMs and propose a slightly modified version of n-gram NNLMs that aims at quantitatively estimating the influence of context words both in terms of their position and their part-of-speech information. We then propose a comparison between the feed-forward and recurrent NNLMs to assess whether long range dependencies have a significant impact on statistical language modeling, considering history sizes ranging from 3 words to an unbounded number of words (in a recurrent architecture). To make this comparison fair, we introduce an extension of the SOUL model that approximates the recurrent architecture with a limited history. With this extension, the associated training procedure can benefit from all the speed-ups and tricks of standard feed-forward NNLMs (e.g., mini-batch and resampling), that make it able to handle large training corpora. Furthermore, we show that this approximation can be effectively used to bootstrap the training of a "true" recurrent architecture. The experimental setup is based on a large scale machine translation task. An abridged version of this chapter has been presented as (Le, Allauzen, and Yvon, 2012b).

The remaining of this chapter is structured as follows. First, Section 5.1 is devoted to present a measure of the influence of each context position in the prediction based on a max variation of NNLMs. Then, in Section 5.2, we analyze the difference between n-gram and recurrent architectures for NNLMs. Based on the discussion on efficient issues, a new structure, pseudo RNNLMs, is introduced in order to speed-up the training procedure for RNNLMs. These types of NNLMs are finally empirically compared in a large scale machine translation framework.

5.1 The usefulness of remote words

In principle, as described in Section 1.4.1.1, the architecture of n-gram feed-forward NNLMs can be divided into three parts. The first one is the input part (ending at the input layer) which aims at representing the context of the prediction in a continuous space. The second one is the hidden part which consists of several nonlinear layers. The output part is a set of softmax layers which compute the probability of all possible successor words given the context. In this section, we will propose a new type of NNLM based on the modification of the hidden part. This new type is then used to measure the importance of context words.

It is worth noticing that to make the later comparison between n-gram and recurrent NNLMs clearer, hereafter, we change the notation of n-grams, from w_1^n to $w_{-(n-1)}^{-1}w$, so the history w_1^{n-1} becomes $w_{-(n-1)}^{-1}$, w_i represents the i^{th} previous word and the predicted word w_n becomes w. Therefore, the role of NNLMs is to estimate the probability $P(w|w_{-(n-1)}^{-1})$. As RNNLMs do not follow the n-gram assumption and can make use of all history words, their role is to estimate the probability $P(w|w_{-\text{inf}}^{-1})$, where $w_{-\text{inf}}$ is misused to represent the first word of the context.

5.1.1 Max NNLMs

In an n-gram feed-forward NNLM, the $n-1$ previous words are projected in the shared continuous space and their representations are then concatenated to form a single input vector **i**, as illustrated in the left part of Figure 5.1:

$$\mathbf{i} = \{\mathbf{R}^T\mathbf{v}_{-(\mathbf{n}-\mathbf{1})}; \mathbf{R}^T\mathbf{v}_{-(\mathbf{n}-\mathbf{2})}; \ldots; \mathbf{R}^T\mathbf{v}_{-\mathbf{1}}\}, \tag{5.1}$$

where $\mathbf{v}_{-i} \in \mathbb{R}^{\mathcal{V}}$ is the 1-of-$\mathcal{V}$ coding vector of the i^{th} previous word and $\mathbf{R} \in \mathbb{R}^{\mathcal{V}\times M}$, the projection matrix. A nonlinear transformation is then applied to compute the first hidden layer **h** as follows:

$$\mathbf{h} = f\left(\mathbf{W}^{\mathbf{h}}\mathbf{i} + \mathbf{b}\right), \tag{5.2}$$

with f stands for the nonlinear function and $\mathbf{W}^{\mathbf{h}} \in \mathbb{R}^{H\times(n-1)M}$. Note that in general, we can use more than one hidden layer.

Conventional n-gram back-off LMs are usually limited to small values of n, and using n greater that 4 or 5 does not seem to be of much use. Indeed, previous experiments using very large speech recognition systems indicated that the gain obtained by increasing the n-gram order from 4 to 5 is almost negligible, whereas the model size increases drastically. While using long context seems to be very impractical with back-off LMs, the situation is quite different for NNLMs due to their specific architecture. In fact, increasing the context length for an NNLM mainly implies to expend the projection layer with one supplementary projection vector, which can furthermore be computed very easily through a look-up operation. The overall complexity of NNLMs thus only grows linearly with n in the worst case (Schwenk, 2007).

In order to better investigate the impact of each context position in the prediction, we introduce a slight modification of this architecture in a manner analog to the proposal of Collobert and Weston (2008) (see Section 1.4.1.5 for more details). In this variation, the computation of the hidden layer defined by Equation (5.2) is replaced by:

$$\mathbf{h} = f\left(\max_{i}\left[\mathbf{W}_{\mathbf{i}}^{\mathbf{h}}\mathbf{R}^T\mathbf{v}_{-\mathbf{i}}\right] + \mathbf{b}\right), \tag{5.3}$$

where $\mathbf{W}_{\mathbf{i}}^{\mathbf{h}} \in \mathbb{R}^{H\times M}$ is the submatrix of $\mathbf{W}^{\mathbf{h}} \in \mathbb{R}^{H\times(n-1)M}$ comprising the columns related to the i^{th} history word, and the max function is to be understood component-wise. The transpose of a product $\mathbf{W}_{\mathbf{i}}^{\mathbf{h}}\mathbf{R}^T \in \mathbb{R}^{H\times\mathcal{V}}$ can then be considered as defining the projection space for the i^{th} position. After the projection of all the context words into these spaces, the max function selects,

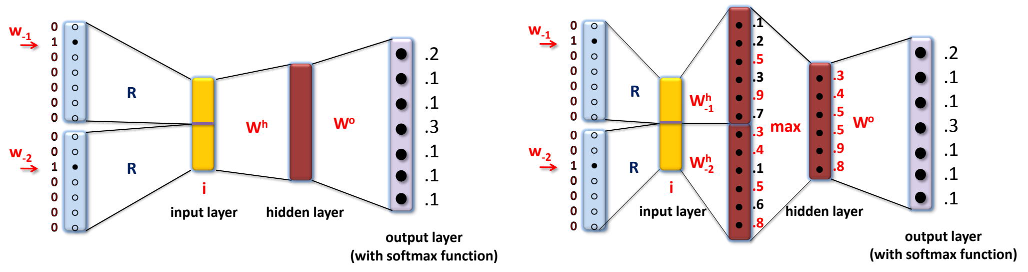


Figure 5.1: *Standard and max neural network language model architectures. After max operation, at the hidden layer, for each element (node), the value (in red) is selected (the bias vector is ignored for simplification).*

for each dimension k, among the $n-1$ values ($[\mathbf{W}^{\mathbf{h}}_{\mathbf{i}}\mathbf{R}^T\mathbf{v}_{-\mathbf{i}}]_k$) the most active one, which we also assume to be the most relevant value for the prediction.

More formally, the selection vector **a** is computed using the component-wise argmax as follows:

$$\mathbf{a} = \operatorname*{argmax}_{i} \left[\mathbf{W}^{\mathbf{h}}_{\mathbf{i}}\mathbf{R}^T\mathbf{v}_{-\mathbf{i}}\right] \tag{5.4}$$

Then, by counting the number of times the index of each position occurs in this vector, we can rank the context words in terms of their importance to the prediction. For a sake of clarify, the new architecture is depicted in the right part of Figure 5.1.

Note that in (Collobert and Weston, 2008), the max hidden layer was proposed mainly to model a variable number of input features. Our motivation for using this variant is different: we mostly aim to analyze the influence of context words based on the selection rates implicitly performed through this function.

5.1.2 Experimental evaluation

We now turn to the experimental part, starting with a description of the experimental setup. We will then present an attempt to quantify the relative importance of history words in terms of position and POS-tag information.

The tasks considered in our experiments are derived from the shared translation track of WMT 2011 (translation from English to French), the same as in Section 2.4.2 where SOUL NNLMs were evaluated. For language model, the training data consist of 2.5 billion words with a vocabulary of about 500k words. Throughout this study, we will consider a 10-gram max NNLM. It is trained following the enhanced SOUL training scheme as described in Chapter 2: the dimension of the projection word space is 500; the size of two hidden layers are respectively 1000 and 500; the shortlist contains 2000 words; and the nonlinearity is introduced with the sigmoid function. Resampling rates are set in order to guarantee that after selecting, there are 75% of in-domain data (monolingual News data 2008-2011) and 25% of the other data. At each epoch, the model parameters are updated using approximately 50 million words for the last training step and about 140 million words for the previous ones. All parameters are shown in the left column of Table 5.1.

parameter	SOUL NNLM	(pseudo and true) RNNLM
training data size	$\approx 2500M$	
vocabulary size	$\approx 500{,}000$	
number of models	1	
order (n)	4, 6, 8, 10	10 and $BPTT = 9$
projection space (M)	500	
hidden layer size (H)	$1000 - 500$	
nonlinear activation	sigmoid	
shortlist size	2000	
number of top classes	2000	
number of examples/epoch (shortlist)	$\approx 140M$	
number of examples/epoch (OOS)	$\approx 140M$	
number of examples/epoch (all words)	$\approx 50M$	
number of epochs (shortlist)	2	
number of epochs (OOS)	2	
number of epochs (all words)	8	6 and 3
learning rate	1×10^{-2}	5×10^{-2} and 2×10^{-3}
learning rate decay	not used	
weight decay	3×10^{-5}	
block size	128	32

Table 5.1: *NNLM configurations. In the second column, in case of having two parameters in one row, the first is for the pseudo RNNLM, the second is for the true RNNLM. Note that learning rates are different because they are empirically tuned for each type of model.*

We first analyze the influence of each context word with respect to their distance from the predicted word and to their POS tag. The quantitative analysis relies on the variant of the n-gram architecture based on Equation (5.3). Using this architecture, it is possible to keep track of the most important context word for each prediction.

Figure 5.2 represents the selection rate with respect to the word position in the context: on this graph, we represent the percentage of coordinates in the input layer that are selected for each position in the history. As expected, the influence of a context word decreases with its position, with the previous word being by large the most important one. Very remote words (at a distance between 7 and 9) have almost the same, weak, influence, with a selection rate around 2.5%. This is consistent with the perplexity results of n-gram NNLMs as a function of n, which are reported in Table 5.3: the difference between all orders from 4-gram to 8-gram are significant, while the difference between 8-gram and 10-gram is negligible.

This result goes in line with the conclusions in (Rosenfeld, 1994) where

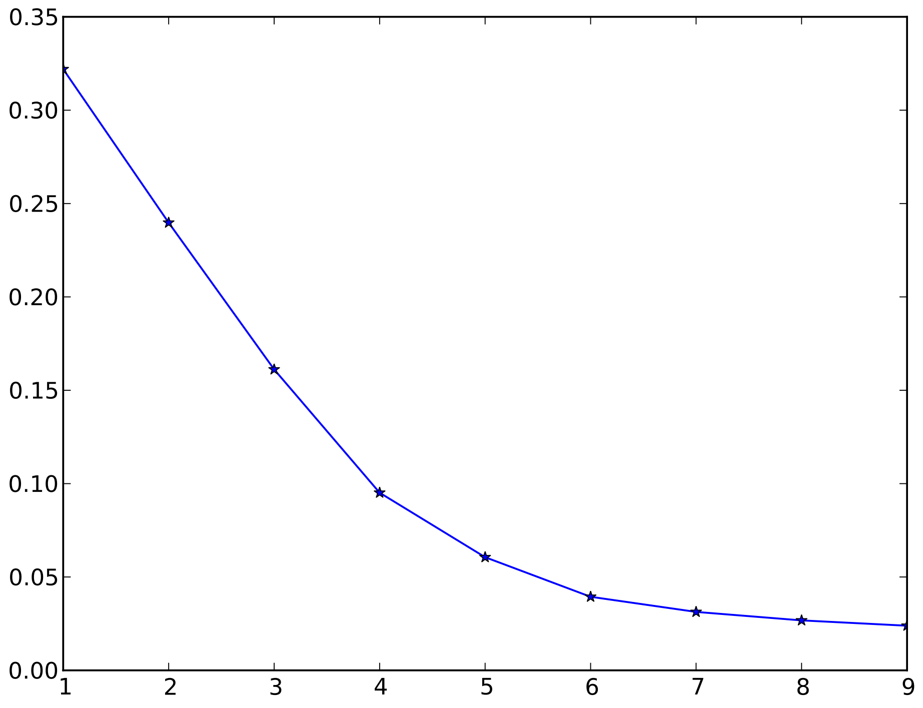


Figure 5.2: *Average selection rate per word position for the* max*-based NNLM, computed on the test set (newstest2009,2010,2011). On x-axis, the number k represents the k^{th} previous word.*

the authors tried to measure the average mutual information between w_i and word w_{i-d} in the Brown Corpus: *"As expected, we found perplexity to be low for $d = 1$, and to increase significantly as we moved through $d = 2, 3, 4$ and 5. For $d = 6 \ldots 10$, training-set perplexity remained at about the same level. . . . We concluded that significant information exists in the last 5 words of the history."* (page 16). The main difference is that in this approach, the influence of each context word was measured separately from the other, we instead, use solely one model to measure all influences at once.

We now analyze the influence by main syntactic category. The POS-tag information is generated using the TreeTagger (Schmid, 1994); subtypes of a main tag are merged together so as to reduce the total number of categories. For example, all the tags for verbs are merged into the same VER class. Adding the special token <s> (start of sentence), the final tagset contains 17 tags (see Table 5.2).

The average selection rates for each tag are listed in Figure 5.3: for each category, we display (in bars) the average number of components that correspond to a word in that category when this word is in previous position. Rare tags (INT, ABK, ABR and SENT) seem to provide a very useful information and have very high selection rates. Conversely, DET, PUN and PRP words occur relatively frequently and belong to the less selective group. The two most frequent tags (NOM and VER) have a medium selection rate (approximately 0.5).

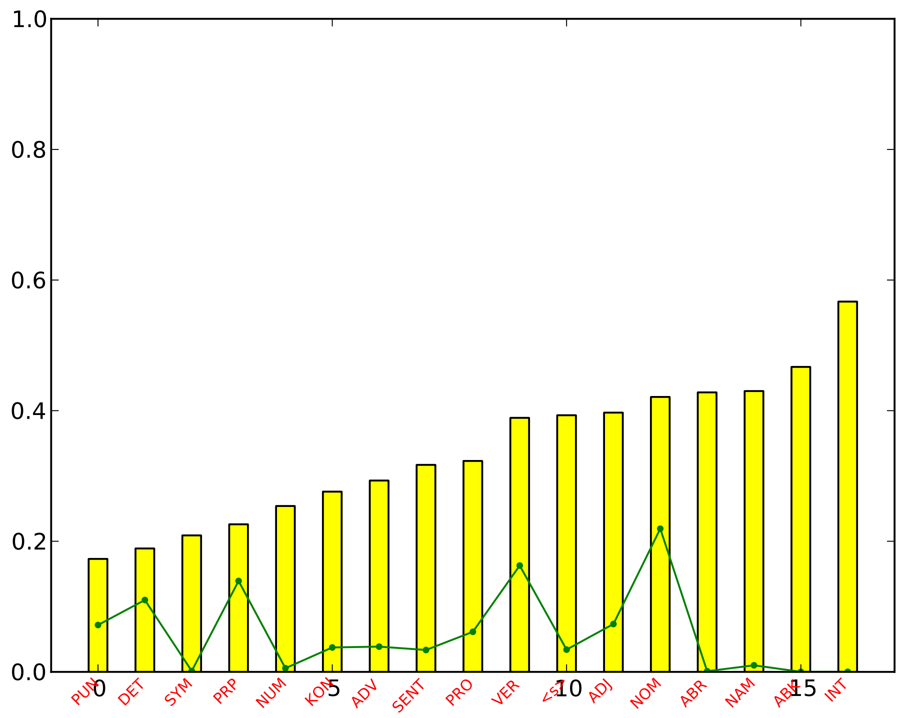


Figure 5.3: *Average selection rate of max function of the first previous word in terms of word POS-tag information, computed on the test set (newstest2009,2010,2011). The green line represents the distribution of occurrences of each tag.*

5.2 N-gram and recurrent NNLMs in comparison

In this section, we are going to present a different view on RNNLM structures so as to clarify the main difference between standard feed-forward (or n-gram) and recurrent NNLMs. Pseudo RNNLMs are then proposed and introduced in the training algorithm for RNNLMs. Experimental results on a large scale machine translation task are finally reported.

5.2.1 Pseudo RNNLMs

As described in Section 5.1, the architecture of n-gram feed-forward NNLMs consist of the input, the hidden and the output parts. For RNNLMs, following

tag	meaning	example
ABR	abreviation	etc FC FMI
ABK	other abreviation	ONG BCE CE
ADJ	adjective	officielles alimentaire mondial
ADV	adverb	contrairement assez alors
DET	article;	une les la
	possessive pronoun	ma ta
INT	interjection	oui adieu tic-tac
KON	conjunction	que et comme
NAM	proper name	Javier Mercure Pauline
NOM	noun	surprise inflation crise
NUM	numeral	deux cent premier
PRO	pronoun	cette il je
PRP	preposition;	de en dans
	preposition plus article	au du aux des
PUN	punctuation;	: , -
	punctuation citation	"
SENT	sentence tag	? . !
SYM	symbol	%
VER	verb	ont fasse parlent
<s>	start of sentence	

Table 5.2: *List of grouped tags from TreeTagger.*

their description in Section 1.4.1.4, we need to specify more formally the input layer in order to make a comparison with the standard approach. So let us formalize here the definition of the input layer:

The input layer is the first layer of the neural network language model which can be considered as a function of all continuous representation of the context words.[1]

Recurrent networks are designed to recursively handle an arbitrary number of context words. Their hidden layer is computed as follows:

$$\mathbf{h} = f(\mathbf{W}\mathbf{h}_{-1} + \mathbf{R}^T\mathbf{v}_{-1})$$

In Figure 5.4, their original architecture is depicted in the top right, to be compared to the standard n-gram architecture, in the top left. Thanks to the unfolded representation (the botom right of Figure 5.4), which is also described in detail in Section 1.4.1.4, we can see that the hidden layer $\mathbf{h}$ can also be considered as the input layer with respect to the above definition. To predict

[1]This means that at the input layer, the projection (or look-up) of context words in the continuous space has been already computed.

the word w, $\mathbf{h}$ is constructed as the function of continuous representations of all words in the context (from the first word of the current sentence <s> to w_{-1}). Note that originally, the context of RNNLMs can contain all past words, up to the first word of the document but we restrict it here to the current sentence, so as to make our later approximation feasible. In general, in RNNLMs, we can also add some layers between $\mathbf{h}$ and the output layer. In this case, their structure is divided into three parts as with feed-forward models. The first part is up to $\mathbf{h}$, unfortunately called the input layer from now. The hidden part is formed by the additional layers. The output part is the same, as before a set of softmax layers.

In the bottom left of Figure 5.4, we show our pseudo architecture of RNNLMs that differs from the feed-forward architecture (in the top left) in its input part. We use the same deep architecture to model the relation between the input word presentations and the input layer as in RNNLMs (the part in the rectangle of the unfolded representation in this figure). However, we explicitly restrict the context to the $n-1$ previous words. For instance, in this figure, while the input layer ($\mathbf{h}$) of a RNNLM takes all words in the context (from <s> to w_{-1}), the trigram approximation is restricted to take only words w_{-2} and w_{-1} into account.

5.2.2 Efficiency issues

Training a standard NNLM can be achieved by maximizing the log-likelihood of the parameters on some training corpus following the stochastic back-propagation (SBP) algorithm as suggested in (Bengio, Ducharme, and Vincent, 2000). RNNLMs are usually trained using a variant of SBP called the *Back-Propagation Through Time* (BPTT) (Rumelhart, Hinton, and Williams, 1986; Mikolov et al., 2011b). In Section 1.4.2, we described a general way of training NNLMs. For feed-forward NNLMs, there exist some methods that can drastically reduce the overall training time, reducing from weeks to days. However, adapting these methods to RNNLMs is not straightforward. An efficient training scheme for SOUL NNLMs was introduced in Section 2.3 (Algorithm 3). Adapting it to RNNLMs is not trivial either. Several techniques are therefore proposed in this section to address these issues. Note that some other possible speed-up techniques for RNNLMs were discussed in Section 1.4.1.4. We also apply some of these ideas in a similar way except for the use of a direct connection between the input and output layers, because it hinders the comparison between feed-forward and recurrent NNLMs.

Reducing the training data Our usual approach for training large scale models is based on n-gram level *resampling*. In this approach, a different subset of the training data is used for each training epoch. This is not directly compatible with RNNLMs as usual RNNLMs make use of the entire context, from the current word position all the way back to the beginning of the document, meaning that the contexts have to be presented in the right order. However,

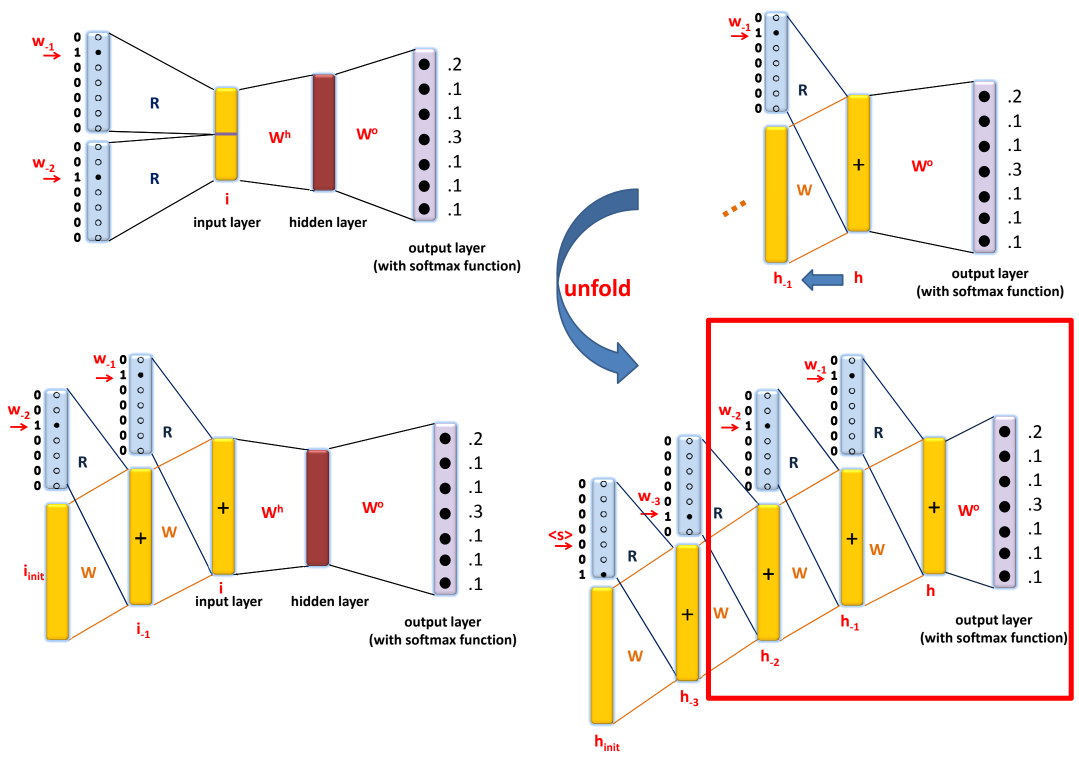


Figure 5.4: *Approximation of the RNNLM to the feed-forward NNLM in case of trigram. Top left: Feed-forward trigram NNLM; top right: RNNLM; bottom right: Unfolded RNNLM; bottom left: Pseudo RNNLM. The pseudo RNNLM is obtained by copying only the part in the rectangle of the unfolded RNNLM, then adding one hidden layer. In the experiments below, we use long-context models with* 10*-gram.*

by restricting the context to sentences, data resampling can be carried out at the sentence level. This means that the input layer is reinitialized at the beginning of each sentence so as to "forget" the memory of the previous sentences. A similar proposal is made in (Mikolov et al., 2011c), where temporal dependencies are limited to the level of paragraph. Another useful trick, which is also adopted here, is to use different sampling rates for the various subparts of the data, thus boosting the use of in-domain versus out-of-domain data.

Bunch mode Bunch mode training processes sentences by batches of several examples, thus enabling matrix operation that are performed very efficiently by existing numerical library such as BLAS. After resampling, the training data is divided into several sentence flows which are processed simultaneously. While the number of examples per batch can be as high as 128 without any visible loss of performance for n-gram NNLMs, we found, after some preliminary experiments, that the value of 32 seems to yield a good trade-off between the computing time and the performance for RNNLMs. Using such batches, the training time can be reduced by a factor of 8 at the price of a slight loss (less than 2%) in perplexity.

SOUL training scheme The enhanced SOUL training scheme (Algorithm 3) integrates several steps used to deal with the fact that the output vocabulary is split in two subparts: very frequent words are in the so-called *shortlist*, and are handled differently from the less frequent ones. This setting cannot be easily reproduced with RNNLMs because it requires to first consider a restricted output vocabulary (the shortlist at Step 1 and out-of-shortlist (OOS) words at Step 3), that is then extended to include the complete prediction vocabulary. In practice, we need to efficiently train a model which takes as input the full vocabulary and as output a different vocabulary. Let us consider the following training sentence:

It is the practical ***guide*** *for the* ***army*** *always to heed the directions of the* ***party***

In this example, we assume that the words in bold are not in the output vocabulary. We process this sentence from the beginning without any problem until we meet the out of output vocabulary word **guide**. We cannot update the parameters with this example but we need to keep going on, so as not to waste the remaining examples. As a result, we have to ignore the update (the backward pass) but nevertheless doing the forward pass in order to have new values of the model parameters for a new example, meaning that we have to do 3 additional forward passes for this sentence. During Step 1 of the SOUL training scheme, the output vocabulary contains only in-shortlist words. Therefore, the number of "wasted" forward passes is equal to the number of OOS words in the training data. As the OOS words are less frequent words, this quantity does not slow down significantly the training procedure. On the contrary, at Step 3, the output vocabulary only contains OOS words, so the training time is drastically increased because now, it is equal to the number of shortlist words in the training data, which is very large as a consequence of the Zipf's law (Zipf, 1932). Moreover, from a practical point of view, by resampling at the sentence level, at Step 1 and Step 3, instead of keeping only useful n-grams into memory, we have to handle whole sentences that contain also useless n-grams. It therefore increases the size of resampling data which can then cause memory problems.

By contrast, by bounding the recurrence to a dozen or so previous words, we obtain an n-gram version of RNNLMs (the pseudo architecture introduced in Section 5.2.1). The SOUL training scheme can be adopted for this new kind of model. After that, the parameter values of the final pseudo model are then plugged into a truly recurrent architecture. We then train the true RNNLM for several epochs until convergence. The proposed training procedure is summarized in Algorithm 6. This model is finally used for inference. It is worth noticing that our proposed pseudo recurrent architecture is just a convenient intermediate model that is used to efficiently train a RNNLM. Intuitively, when n is large, the pseudo RNNLM is considered as a good approximate version of a RNNLM. In the light of the results reported below, we content ourselves with $n = 10$.

Algorithm 6 Recurrent training scheme

```
**Step 1, shortlist pre-training** provides a first estimate of the continuous space by training a shortlist pseudo RNNLM.
**Step 2, word clustering** derives a clustering tree from the information in the continuous space.
**Step 3, OOS pre-training** trains a pseudo RNNLM with OOS words as output.
**Step 4, full pseudo training** trains a pseudo RNNLM with the tree structure induced in Step 2 and with the parameters initialized using the parameters obtained at Step 3 and Step 1.
**Step 5, full recurrent training** Use the parameters of a pseudo RNNLM to initialize the parameters of a true RNNLM. Train this model until convergence
```

5.2.3 MT Experimental evaluation

The same setup is used as that presented in Section 5.1.2. Note that the robust baseline target n-gram language model is built as the interpolation of submodels LMs trained on different parts of the training data. All standard n-gram NNLMs are trained with the same configuration parameters as with the max NNLM. For the RNNLM, the parameter that limits the back-propagation of errors through time was set to 9 (see (Mikolov et al., 2010) for details). This parameter can be considered to play a role that is similar to the history size in our pseudo RNNLM: a value of 9 in the recurrent setting is equivalent to $n = 10$. All configuration parameters are listed in Table 5.1.

For the machine translation task, as usual, we carry out m-best rescoring: the NNLMs probability of each hypothesis was computed and the list was accordingly reordered. The NNLM weights were optimized as the other feature weights using MERT (Och, 2003). For all our experiments, the size of the lists is 300.

To clarify the impact of the language model order in translation performance, we consider three different ways to use NNLMs. In the first setting, the NNLM is used *alone* and all the scores provided by the MT system are ignored. In the second setting (*replace*), the NNLM score replace the score of the standard back-off LM. Finally, the score of the NNLM can be added in the linear combination (*add*). In the last two settings, the weights used for m-best reranking are retuned with MERT.

model	**ppl**	**BLEU**		
		alone	*replace*	*add*
baseline	90	29.4	31.3	-
4-gram	92	29.8	31.1	31.5
6-gram	82	30.2	31.6	**31.8**
8-gram	78	**30.6**	31.6	**31.8**
10-gram	**77**	30.5	**31.7**	**31.8**
recurrent	81	30.4	31.6	**31.8**

Table 5.3: *Results for the English to French task obtained with the baseline system and with various NNLMs. Perplexity (ppl) is computed on newstest2009,2010,2011 while BLEU is on the test set (newstest2010).*

Table 5.3 summarizes the BLEU scores obtained on the *newstest2010* test set. BLEU improvements are observed with feed-forward NNLMs using a value of $n = 8$ with respect to the baseline ($n = 4$). A further increase from 8 to 10 only provides a very small BLEU improvement. These results strengthen the assumption made in Section 5.2.2, there seems to be very little information in remote words (above $n = 8$)[2]. It is also interesting to see that the 4-gram NNLM achieves a comparable perplexity to the conventional 4-gram model, yet delivers a small BLEU increase in the *alone* condition.

Surprisingly[3], on this task, recurrent models seem to be comparable with 8-gram NNLMs which use a shorter context. The reason may be the deep architecture of recurrent model that makes it hard to be trained on a large scale task. Nevertheless, it is proved that with the pseudo recurrent model, it is feasible to train a recurrent model on a large task. With 10% of perplexity reduction as compared to a back-off model, it yields comparable performances as the one described in (Mikolov et al., 2011b), despite having a smaller number of parameters (see Table 5.4). To the best of our knowledge, it is the first

[2]At least for these data.

[3]Personal communication with T. Mikolov: on the Wall-Street Journal data set (a small task), the recurrent model described in (Mikolov et al., 2011c) outperforms the 10-gram NNLM.

time that a RNNLM is trained on such large data set (2.5 billion words) in a reasonable time (about 11 days).

model	**number of parameters**
baseline	1,159,079,500
4-gram NNLM	496,225,231
6-gram NNLM	497,225,231
8-gram NNLM	498,225,231
10-gram NNLM	499,225,231
RNNLM	495,475,231

Table 5.4: *The number of parameters of language models.*

5.3 Summary

In this chapter, we have investigated several types of NNLMs along with conventional LM in order to assess the influence of long range dependencies in the language modeling task: from recurrent model that can recursively handle an arbitrary number of context words to n-gram NNLMs with n varying between 4 and 10.

Our contributions are two-fold. First, by using the new methodology of NNLMs with max hidden layer, experimental results with the data derived from the WMT 2011 machine translation task show that the influence of word further than 9 can be neglected. Therefore, the n-gram assumption with $n \approx 10$ appears to be well-founded. Then, by introducing the pseudo RNNLM and restricting the context of RNNLMs to the current sentence, RNNLMs can benefit from the advanced training schemes and their training time can be divided by a factor 8 without significant loss on the performances. We compared n-gram NNLMs and RNNLMs within a large scale MT task, with monolingual data containing $\approx$ 2.5 billion words. Experimental results show that using long range dependencies ($n = 10$) with a SOUL language model significantly outperforms conventional LMs. In this setting, the use of a recurrent neural architecture does not yield any improvements over an n-gram neural architecture, both in terms of perplexity and BLEU.

Our conclusion is that the main issue of the n-gram model does not seem to be the conditional independence assumptions, but the use of too small values for n. All these results also suggest that in the future, novel methods for language modeling must take into account long range dependencies.

Part III

Continuous Space Neural Network Translation Models

Chapter 6

Continuous Space Neural Network Translation Models

Contents

The unreliability of conventional maximum likelihood estimates hinders the performance of existing phrase-based translation models which nevertheless remain the state-of-the-art approach in machine translation. For lack of sufficient training data, most models only consider a small amount of context. To address this data sparsity issues faced by translation model, several remedies were proposed in the literature. Smoothing is obviously one possibility (Foster, Kuhn, and Johnson, 2006). Another one is to use *factored language models*, introduced in (Bilmes and Kirchhoff, 2003), then adapted for translation models in (Koehn and Hoang, 2007; Crego and Yvon, 2010). Such approaches require using external linguistic analysis tools which are error prone; moreover, they did not seem to bring clear improvements, even when translating into morphologically rich languages.

Following a different direction, there exist also some attempts that try to use continuous spaces on machine translation. To the best of our knowledge, the first work is (Schwenk, R. Costa-jussa, and R. Fonollosa, 2007), where the authors introduced the model referred here to as the standard *n*-gram translation model in Section 6.2.1. This model is a bilingual extension of the continuous space language model studied in the previous chapters where the basic unit is the tuple (or equivalently the phrase pair). The resulting vocabulary being too large to be handled by neural networks without a structured output layer, the authors had thus to restrict the set of the predicted units to a 8*k* shortlist. Moreover, in (Zamora-Martinez, Castro-Bleda, and Schwenk, 2010), the authors propose a tighter integration of a continuous space model within an *n*-gram based MT approach both for the tuple and target LMs. A different approach, described in (Sarikaya et al., 2008), divides the problem in two parts: first the continuous representation is obtained by an adaptation of the Latent Semantic Analysis (LSA) (Deerwester et al., 1990); then a Gaussian mixture model is learned using this continuous representation and included in a hidden Markov model. One problem with this approach is the separation between the training of the continuous representation on the one hand, and the training of the translation model on the other hand.

Based on the same idea, we explore several continuous space translation models, where translation probabilities are estimated using a continuous representation of the translation units in lieu of standard discrete representations. In order to handle a large set of translation units, these representations and the associated estimates are jointly computed using a multi-layer neural network and the SOUL architecture. In small scale and large scale machine translation experiments, we will show that the resulting models can effectively be trained and used on top of a phrased based translation system, delivering significant improvements in performance. For reference, most of the work was published in (Le, Allauzen, and Yvon, 2012a).

The content of this chapter is structured as follows. In Section 6.1, we briefly describe a phrase-based statistical machine translation system. After that, several variations of translation models are presented and analyzed in Section 6.2. The experimental results for both small and large scale tasks are finally reported in Section 6.3.

6.1 Phrase-based statistical machine translation

The phrase-based approach to statistical machine translation (SMT) is based on the following inference rule, which, given a source sentence **s**, selects the target sentence **t** and the underlying alignment **a** maximizing the following term:

$$P(\mathbf{t}, \mathbf{a}|\mathbf{s}) = \frac{1}{Z(\mathbf{s})} \exp\left(\sum_{i=1}^{F} \lambda_i f_i(\mathbf{s}, \mathbf{t}, \mathbf{a})\right), \tag{6.1}$$

where F feature functions (f_i) are weighted by a set of coefficients (λ_i), and Z is a normalizing factor. The phrase-based approach differs from other approaches by the hidden variables of the translation process: the segmentation of a parallel sentence pair into phrase pairs and the associated phrase alignments.

This formulation was introduced in (Zens, Och, and Ney, 2002) as an extension of the word based models (Brown et al., 1993), then later motivated within a discriminative framework (Och and Ney, 2004). One motivation for integrating more feature functions was to improve the estimation of the translation model $P(\mathbf{t}|\mathbf{s})$, which was initially based on relative frequencies, thus yielding poor estimates.

This is because the units of phrase-based models are *phrase pairs*, made of a source and a target phrase; such units are viewed as the events of discrete random variables. The resulting representations of phrases (or words) thus entirely ignore the morphological, syntactic and semantic relationships that exist among those units in both languages. This lack of structure hinders the generalization power of the model and reduces its ability to adapt to other domains. Another consequence is that phrase-based models usually consider a very restricted context[1].

This is a general issue in statistical NLP and many possible remedies have been proposed in the literature. They are often inherited directly from the methods for language modeling which are described in Chapter 1, such as, the way of using smoothing techniques, of introducing linguistically enriched, or more abstract, representations of words, phrases. Among them, the technique based on the representation in a continuous space with neural networks is not an exception. In the context of SMT, (Schwenk, R. Costa-jussa, and R. Fonollosa, 2007) is the first attempt to estimate translation probabilities in a continuous space. However, because of the proposed shortlist based neural architecture (see Section 1.4.3.2 for more details), the authors only consider a very restricted set of translation units, and therefore report only a slight impact on translation performance. Our proposed SOUL structure, which is introduced in Chapter 2 seems especially relevant, as it is able, through the use of class-based models, to handle arbitrarily large vocabularies and opens the way to enhanced neural translation models.

In the next sections, we are going to explore various neural architectures for n-gram translation models that consider three different ways to factor the joint probability $P(\mathbf{s}, \mathbf{t})$ differing by the units (respectively phrase pairs, phrases or words) that are projected in continuous spaces. While these decompositions are theoretically straightforward, they were not considered in the past because of data sparsity issues and of the resulting weaknesses of conventional *maximum likelihood* estimates. Our main contribution is then to show that such joint distributions can be efficiently computed by neural networks, even for very long context sizes; and that their use yields significant performance improvements. These models are evaluated in an m-best rescoring step using the framework of phrased based systems.

6.2 Variations on the n-gram approach

In the n-gram based approach (Casacuberta and Vidal, 2004; Mariño et al., 2006; Crego and Mariño, 2006), translation is divided in two steps: a source reordering step and a translation step. Source reordering is based on a set of learned rewrite rules that non-deterministically reorder the input words so as to match the target order thereby generating a lattice of possible reorderings. Translation then amounts to finding the most likely path in this lattice using an

[1]Typically a small number of preceding phrase pairs for the n-gram based approach (Crego and Mariño, 2006), or no context at all, for the standard approach of Koehn et al. (2007).

n-gram translation model[2] of *bilingual units*.

6.2.1 The standard n-gram translation model

N-gram translation models (TMs) rely on a specific decomposition of the joint probability $P(\mathbf{s}, \mathbf{t})$, where $\mathbf{s}$ is a sequence of I reordered source words $(s_1, \ldots, s_I)$ and $\mathbf{t}$ contains J target words $(t_1, \ldots, t_J)$. This sentence pair is further assumed to be decomposed into a sequence of L bilingual units called *tuples* defining a joint segmentation: $(\mathbf{s}, \mathbf{t}) = u_1, \ldots, u_L$. In the approach of Mariño et al. (2006), this segmentation is a by-product of source reordering, and ultimately derives from initial word and phrase alignments. In this framework, the basic translation units are *tuples*, which are the analogous of phrase pairs, and represent a matching $u = (\bar{s}, \bar{t})$ between a source $\bar{s}$ and a target $\bar{t}$ phrase (see Figure 6.1). Using the n-gram assumption, the joint probability of a segmented sentence pair decomposes as:

$$P(\mathbf{s}, \mathbf{t}) = \prod_{i=1}^{L} P(u_i | u_{i-1}, \ldots, u_{i-n+1}) \tag{6.2}$$

A first issue with this model is that the elementary units are bilingual pairs, which means that the underlying vocabulary, hence the number of parameters, can be quite large, even for small translation tasks. Due to data sparsity issues, such models are bound to face severe estimation problems. Another problem with Equation (6.2) is that the source and target sides play symmetric roles, whereas the source side is known and the target side must be predicted.

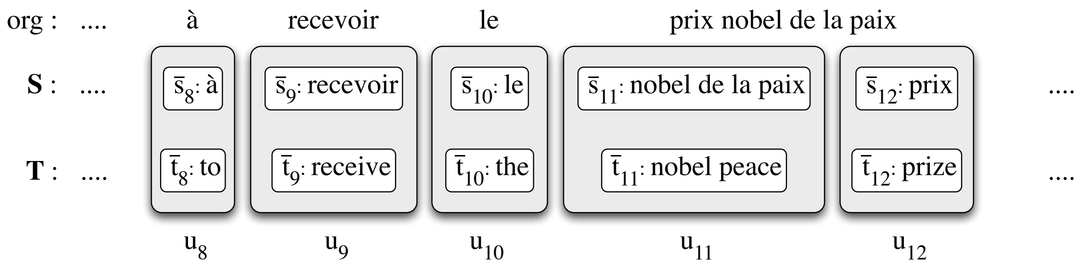


Figure 6.1: *Extract of a French-English sentence pair segmented in bilingual units. The original (org) French sentence appears at the top of the figure, just above the reordered source* $\mathbf{s}$ *and target* $\mathbf{t}$*. The pair* $(\mathbf{s}, \mathbf{t})$ *decomposes into a sequence of L bilingual units (tuples)* $u_1, \ldots, u_L$*. Each tuple* u_i *contains a source and a target phrase:* $\bar{s}_i$ *and* $\bar{t}_i$*.*

[2]Like in the standard phrase-based approach, the translation process also involves additional feature functions that are presented below.

6.2.2 A factored n-gram translation model

To overcome some of these issues, the n-gram probability in Equation (6.2) can be factored by decomposing tuples in two (source and target) parts:

$$\begin{aligned} &P(u_i|u_{i-1},\ldots,u_{i-n+1}) = \\ &\quad P(\bar{t}_i|\bar{s}_i,\bar{s}_{i-1},\bar{t}_{i-1},\ldots,\bar{s}_{i-n+1},\bar{t}_{i-n+1}) \\ &\quad \times P(\bar{s}_i|\bar{s}_{i-1},\bar{t}_{i-1},\ldots,\bar{s}_{i-n+1},\bar{t}_{i-n+1}) \end{aligned} \tag{6.3}$$

Equation (6.3) involves two models: the first term represents a TM, the second term is best viewed as a reordering model since it define a probability for the reordered source sequence. In this formulation, the TM only predicts the target phrase, given its source and target contexts.

Another strength of this formulation is that the elementary events now correspond either to source or to target phrases, but never to pairs of such phrases. The underlying vocabulary is thus obtained as the union, rather than the cross product, of phrase inventories. Finally note that the n-gram probability $P(u_i|u_{i-1},\ldots,u_{i-n+1})$ could also factor as:

$$\begin{aligned} &P(\bar{s}_i|\bar{t}_i,\bar{s}_{i-1},\bar{t}_{i-1},\ldots,\bar{s}_{i-n+1},\bar{t}_{i-n+1}) \\ &\times P(\bar{t}_i|\bar{s}_{i-1},\bar{t}_{i-1},\ldots,\bar{s}_{i-n+1},\bar{t}_{i-n+1}) \end{aligned} \tag{6.4}$$

A formulation that we denote later as an inverse variation factored n-gram TM.

6.2.3 A word factored translation model

A more radical way to address the data sparsity issues is to take (source and target) words as the basic units of the n-gram TM. This may seem to be a step backwards, since the transition from word (Brown et al., 1993) to phrase-based models (Zens, Och, and Ney, 2002) is considered as one of the main recent improvement in MT. One important motivation for considering phrases rather than words was to capture local context in translation and reordering. It should then be stressed that the decomposition of phrases in words is only re-introduced here as a way to mitigate the parameter estimation problems. Translation units are still pairs of *phrases*, derived from a bilingual segmentation in tuples synchronizing the source and target n-gram streams, as defined by Equation (6.3). In fact, the estimation policy described in Chapter 2 will actually allow us to design n-gram models with *longer contexts* than is typically possible in the conventional n-gram approach.

Let s_i^k denote the k^{th} word of source tuple $\bar{s}_i$. Considering again the example of Figure 6.1, s_{11}^1 is to the source word *nobel*, s_{11}^4 is to the source word *paix*, and similarly t_{11}^2 is the target word *peace*. We finally denote $h^{n-1}(t_i^k)$ the sequence made of the $n-1$ words preceding t_i^k in the target sentence: in Figure 6.1,

$h^3(t_{11}^2)$ thus refers to the three context words *receive the nobel* associated with the target word *peace*. Using these notations, Equation (6.3) is rewritten as:

$$\begin{aligned} P(\mathbf{s},\mathbf{t}) = \prod_{i=1}^{L} \Big[\prod_{k=1}^{|\bar{t}_i|} P\big(t_i^k | h^{n-1}(t_i^k), h^{n-1}(s_{i+1}^1)\big) \\ \times \prod_{k=1}^{|\bar{s}_i|} P\big(s_i^k | h^{n-1}(t_i^1), h^{n-1}(s_i^k)\big) \Big] \end{aligned} \tag{6.5}$$

This decomposition relies on the n-gram assumption, this time at the word level. Therefore, this model estimates the joint probability of a sentence pair using two sliding windows of length n, one for each language; however, the moves of these windows remain synchronized by the tuple segmentation. Moreover, the context is not limited to the current phrase, and continues to include words in adjacent phrases. Using the example of Figure 6.1, the contribution of the target phrase $\bar{t}_{11}$ = *nobel, peace* to $P(\mathbf{s},\mathbf{t})$ using a trigram model is:

$$\begin{aligned} &P\big(\text{nobel}|[\text{receive, the}], [\text{la, paix}]\big) \\ \times &P\big(\text{peace}|[\text{the, nobel}], [\text{la, paix}]\big). \end{aligned}$$

Likewise, the contribution of the source phrase $\bar{s}_{11}$ =*nobel, de, la, paix* is:

$$\begin{aligned} &P\big(\text{nobel}|[\text{receive, the}], [\text{recevoir,le}]\big) \\ &\times P\big(\text{de}|[\text{receive, the}], [\text{le,nobel}]\big) \\ &\times P\big(\text{la}|[\text{receive, the}], [\text{nobel, de}]\big) \\ &\times P\big(\text{paix}|[\text{receive, the}], [\text{de,la}]\big). \end{aligned}$$

A positive effect of this new formulation is that the involved vocabularies only contain words, and are thus much smaller. These models are thus less bound to be affected by data sparsity issues. While the TM defined by Equation (6.5) derives from Equation (6.3), an inverse variation can be equivalently derived from Equation (6.4).

6.2.4 Translation modeling with SOUL

In the previous section, we defined three different n-gram translation models, based respectively on Equations (6.2), (6.3) and (6.5). As discussed above, a major issue with such models is to reliably estimate their parameters, the number of which grows exponentially with the order of the model. This problem is aggravated in NLP, due to well-known data sparsity issues. In this work, we take advantage of the neural network architecture for language modeling, the SOUL model, which was introduced in Chapter 2. It thus becomes possible to handle large vocabulary language modeling tasks, a solution that we adapt here to machine translation.

In fact, using the SOUL architecture, it is possible to estimate n-gram distributions for any kind of discrete random variables over, for instance sets of phrases or sets of tuples. The SOUL architecture can thus readily be used as a replacement for the standard n-gram TM described in Section 6.2.1. This is because all the random variables are events over the same set of tuples.

Adopting this architecture for the other n-gram TM respectively described by Equations (6.3) and (6.5) is less straightforward, as they involve two different languages and thus two different vocabularies: the predicted unit is a target phrase (resp. word), whereas the context is made of both source and target phrases (resp. words). A subsequent modification of the SOUL architecture is proposed to make up for "mixed" contexts: rather than projecting all the context words or phrases into the same continuous space (using the matrix $\mathbf{R}$, see Figure 2.1), we use two different projection matrices, one for each language. The input layer is thus composed of two vectors in two different spaces; these two representations are then combined through the hidden layer, the other layers remaining unchanged.

6.3 Experimental evaluation

We now turn to an experimental comparison of the models introduced in Section 6.2. We first describe the tasks and data that are used, before presenting our n-gram based system and baseline setup. Our results are finally presented and discussed. In order to introduce new scores of translation models, again, we resort to a two pass approach: the first pass uses a conventional back-off language model to produce an m-best list; in the second pass, the probability of a SOUL model is computed for each hypothesis, added as a new feature and the m-best list is accordingly reordered[3]. In all the following experiments, we used 10 as the fixed order of SOUL models, and used 300 as the size of m-best list.

6.3.1 Tasks and corpora

The two tasks considered in our experiments are adapted from the text translation track of IWSLT[4] 2011 from English to French (the "TED" talk task): a *small data* scenario where the only training data is a small in-domain corpus; and a large scale condition using all the available training data. Here, we only provide a short overview of the task; all the necessary details regarding this evaluation campaign are on the official website[5].

The in-domain training data consists of 107,058 sentence pairs, whereas for the large scale task, all the data available for the WMT 2011 evaluation (the same used to evaluate SOUL NNLMs in Section 2.4.2) are added. For the latter

[3]The probability estimated with the SOUL model is added as a new feature to the score of an hypothesis given by Equation (6.1). The coefficients are retuned before the reranking step.

[4]International Workshop on Spoken Language Translation

[5]`http://iwslt2011.org`

task, the parallel data includes a large Web corpus, referred to as the GigaWord parallel corpus. This corpus is very noisy and is accordingly filtered using a simple perplexity criterion as explained in (Allauzen et al., 2011).

The total amount of training data is approximately 11.5 million sentence pairs for the bilingual part, and about 2.5 billions of words for the monolingual part. As the provided validation data is quite small, development and test set are inverted, and we finally use a validation set of 1664 sentences, and a test set of 934 sentences. Table 6.1 provides the sizes of the different vocabularies. The n-gram TMs are estimated over a training corpus composed of tuple sequences. Tuples are extracted from the word-aligned parallel data[6] in such a way that a unique segmentation of the bilingual corpus is achieved, allowing to directly estimate bilingual n-gram models (see (Crego and Mariño, 2006) for details).

model	**vocabulary size**			
	small task		*large task*	
	src	*trg*	*src*	*trg*
standard	317k		8847k	
phrase factored	96k	131k	4262k	3972k
word factored	45k	53k	505k	492k

Table 6.1: *Vocabulary sizes for the English to French tasks obtained with various SOUL translation (TM) models. For the factored models, sizes are indicated for both source (src) and target (trg) sides.*

6.3.2 N-gram based translation system

The n-gram based system used here is based on an open source implementation described in (Crego, Yvon, and Mariño, 2011). In a nutshell, the TM is implemented as a stochastic finite-state transducer trained using a n-gram model of (source, target) pairs as described in Section 6.2.1. Training this model requires to reorder source sentences so as to match the target word order. This is performed by a non-deterministic finite-state reordering model, which uses part-of-speech information generated by the TreeTagger (Schmid, 1994) to generalize reordering patterns beyond lexical regularities.

In addition to the TM, fourteen feature functions are included: a *target-language model*; four *lexicon models*; six *lexicalized reordering models* (Tillmann, 2004; Crego, Yvon, and Mariño, 2011); a distance-based *distortion model*; and finally a *word-bonus model* and a *tuple-bonus model*. The four *lexicon models* are similar to the ones used in standard phrase-based systems: two scores correspond to the relative frequencies of the tuples and two lexical weights are estimated from the automatically generated word alignments. The weights

[6]using MGIZA++ (Gao and Vogel, 2008), a multi-threaded version of GIZA++ (Och and Ney, 2003) with default setting.

associated to feature functions are optimally combined using MERT (Och, 2003). All the results in BLEU are obtained as an average of 4 optimization runs[7].

For the small task, the target LM is a standard 4-gram model estimated with the Kneser-Ney discounting scheme interpolated with lower order models (Kneser and Ney, 1995; Chen and Goodman, 1998), while for the large task, the target LM is obtained by linear interpolation of several 4-gram models (see (Lavergne et al., 2011) for details).

As for the TM, all the available parallel corpora are simply pooled together to train a trigram model. Results obtained with this large-scale system are found to be comparable to some of the best official submissions.

6.3.3 Small task evaluation

Table 6.2 summarizes the results obtained with the baseline and different SOUL models, TMs and a target LM. The first comparison concerns the standard n-gram TM, defined by Equation (6.2), when estimated conventionally or as a SOUL model. Adding the latter model yields a slight BLEU improvement of 0.5 point over the baseline. When the SOUL TM is phrased factored as defined in Equation (6.3) the gain is of 0.9 BLEU point instead. This difference can be explained by the smaller vocabularies used in the latter model, and its improved robustness to data sparsity issues. Additional gains are obtained with the word factored TM defined by Equation (6.5): a BLEU improvement of 0.8 point over the phrase factored TM and of 1.7 point over the baseline are respectively achieved. We assume that the observed improvements can be explained by the joint effect of a better smoothing.

model	**BLEU**	
	dev	*test*
baseline	31.4	25.8
adding a SOUL model		
standard TM	32.0	26.3
phrase factored TM	32.7	26.7
word factored TM	33.6	**27.5**
target LM	32.6	26.9

Table 6.2: *Results for the small English to French task obtained with the baseline system and with various SOUL translation (TM) or target language (LM) models.*

The comparison to the condition where we only use a SOUL target LM is interesting as well. Here, the use of the word factored TM still yields to a 0.6 BLEU improvement. This result shows that there is an actual benefit in smoothing the TM estimates, rather than only focus on the LM estimates.

Table 6.3 reports a comparison among the different components and variations of the word factored TM. In the upper part of the table, the model defined

[7]The standard deviations are below 0.1 and thus omitted in the reported results.

model	**BLEU**	
	dev	*test*
adding a SOUL model		
+ $P\big(t_i^k \| h^{n-1}(t_i^k), h^{n-1}(s_{i+1}^1)\big)$	32.6	**26.9**
+ $P\big(s_i^k \| h^{n-1}(t_i^1), h^{n-1}(s_i^k)\big)$	32.1	26.2
+ the combination of both	33.2	27.5
+ $P\big(s_i^k \| h^{n-1}(s_i^k), h^{n-1}(t_{i+1}^1)\big)$	31.7	26.1
+ $P\big(t_i^k \| h^{n-1}(s_i^1), h^{n-1}(t_i^k)\big)$	32.7	**26.8**
+ the combination of both	33.4	27.2

Table 6.3: *Comparison of the different components and variations of the word factored translation model.*

by Equation (6.5) is evaluated component by component: the translation term $P\big(t_i^k|h^{n-1}(t_i^k), h^{n-1}(s_{i+1}^1)\big)$, its distortion counterpart $P\big(s_i^k|h^{n-1}(t_i^1), h^{n-1}(s_i^k)\big)$ and finally their combination, which yields the joint probability of the sentence pair. Here, we observe that the best improvement is obtained with the translation term, which is 0.7 BLEU point better than the "distortion" term. Moreover, the use of just a translation term only yields a BLEU score equal to the one obtained with the SOUL target LM, and its combination with the distortion term is decisive to attain the additional gain of 0.6 BLEU point. The lower part of the table provides the same comparison, but for the variation of the word factored TM. Besides a similar trend, we observe that this variation delivers slightly lower results. This can be explained by the restricted context used by the translation term which no longer includes the current source phrase or word.

6.3.4 Large task evaluations

IWSLT 2011 For the large-scale setting, the training material increases drastically with the use of the additional out-of-domain data for the baseline models. Results are summarized in Table 6.4. The first observation is the large increase of BLEU (+2.4 points) for the baseline system over the small-scale baseline. For this task, only the word factored TM is evaluated since it significantly outperforms the others on the small task (see Section 6.3.3).

In a first scenario, we use a word factored TM, trained only on the small in-domain corpus. Even though the training corpus of the baseline TM is one hundred times larger than this small in-domain data, adding the SOUL TM still yields a BLEU increase of 1.2 point[8]. In a second scenario, we increase the training corpus for the SOUL, and include parts of the out-of-domain data (the WMT part). The resulting BLEU score is here slightly worse than the one obtained with just the in-domain TM, yet delivering improved results with the

[8]Note that the in-domain data was already included in the training corpus of the baseline TM.

respect to the baseline.

model	**BLEU**	
	dev	*test*
baseline	33.7	28.2
adding a word factored SOUL TM		
+ in-domain TM	35.2	29.4
+ out-of-domain TM	34.8	29.1
+ out-of-domain adapted TM	35.5	**29.8**
adding a SOUL LM		
+ out-of-domain adapted LM	35.0	29.2

Table 6.4: *Results for the large English to French translation task obtained by adding various SOUL translation and language models (see text for details).*

In a next attempt, we amend the training regime of the neural network. In a first step, we train conventionally a SOUL model using the same out-of-domain parallel data as before. We then "adapt" this model by running five additional epochs of the back-propagation algorithm using the in-domain data. Using this adapted model yields our best results to date with a BLEU improvement of 1.6 points over the baseline results. Moreover, the gains obtained using this simple domain adaptation strategy are respectively of +0.4 and +0.8 BLEU, as compared with the small in-domain model and the large out-of-domain model. These results show that the SOUL TM can scale efficiently and that its structure is well suited for domain adaptation.

WMT 2012 Word factored SOUL translation models along with SOUL language models are also used in LIMSI *n*-gram based translation systems participated on the WMT 2012 evaluation campaign (see (Le et al., 2012a) for more details about this configuration). They are shown to yield significant improvements in all four directions: English-French (en-fr) pair and English-German (en-de) pair. Note that for the first pair, in both directions, LIMSI systems achieve the best results in terms of both BLEU and human evaluation over 15 participating systems (Callison-Burch et al., 2012). These results are summarized in Table 6.5.

Recently, we have tried to use Moses (Koehn et al., 2007) as a baseline system. With the same quantity of data that we use for WMT 2012, word factored SOUL translation models significantly improve the system performance as shown in Table 6.6.

6.4 Summary

We have presented three ways to adopt our neural network architecture for translation modeling. A first contribution was to produce the first large-scale

direction	**system**	**BLEU**	
		newstest2011	*newstest2012**
en2fr	baseline	32.0	28.9
	+ SOUL (TM and LM)	33.4	29.9
fr2en	baseline	30.2	30.4
	+ SOUL (TM and LM)	31.1	31.5
en2de	baseline	15.4	16.0
	+ SOUL (TM and LM)	16.6	17.0
de2en	baseline	21.8	22.9
	+ SOUL (TM and LM)	22.8	23.9

Table 6.5: *Experimental WMT 2012 results in terms of BLEU scores measured on two test sets: newstest2011 and newstest2012. For newstest2012, the scores are those provided by the organizers.*

direction	**system**	**BLEU**
		newstest2011
fr2en	baseline Moses	29.5
	+ SOUL TM	30.4

Table 6.6: *Experimental WMT 2012 results in terms of BLEU scores measured on the test set newstest2011 when Moses is used as baseline system.*

neural translation model, implemented here in the framework of the n-gram based models, taking advantage of a specific hierarchical architecture (SOUL). By considering several decompositions of the joint probability of a sentence pair, several bilingual translation models were presented and discussed.

As it turned out, using a factorization which clearly distinguishes the source and target sides, and only involves word probabilities, proved an effective remedy to data sparsity issues and provided significant improvements over the baseline on the "TED" talk translation English to French tasks, adapted from the IWSLT 2011 evaluation, in both small and large scale settings.

We also investigated various training regimes for these models in a cross domain adaptation setting. Our results show that adapting an out-of-domain SOUL TM is both an effective and very fast way to perform bilingual model adaptation. Adding up all these novelties finally brought us a 1.6 BLEU point improvement.

Moreover, this approach was also experimented within the systems we submitted to the shared translation task of WMT 2012. The achievements in a large scale setup and for different language pairs are in the same ballpark.

Finally, even though our proposed models were originally designed to work only within the framework of n-gram based MT systems, using such models in other systems is straightforward. Introducing them into Moses, another conventional phrase-based system was shown to be also helpful.

Conclusion

Contributions

Having been used for several decades, *n*-gram models with smoothing techniques are still a cornerstone of modern language modeling in NLP systems. After a lot of experiments with these models in a variety of languages, genres, data sets and applications, the vexing conclusion is that these models are still very difficult to improve upon. Many different approaches have been discussed in the literature; yet, very few of them have been shown to deliver consistent performance gains, especially on large scale tasks. The reason behind is that, most of them considers each word as one discrete symbol with no relation with the others. This fact implies that they cannot take into account the similarity between words which is an important way to tackle the widely known data sparsity issue.

Continuous space neural network language models (NNLMs) have been shown to be a promising alternative approach to language modeling by means of their elegant property of treating words in a completely different way. Words are projected in a continuous space and the similarity between them is taken into account as the distance of their representations in this space. Their advantage is that the word representations along with the associated probability estimates are learned discriminatively and automatically in a multi-layer neural network framework. Significant and consistent improvements are observed when this type of model is applied to automatic speech recognition and machine translation tasks. Their major drawback remains the computational complexity, which does not scale well to the huge quantity of data available nowadays. Several speed-up techniques are therefore required (Schwenk, 2007). These techniques, notably the use of shortlist, hinders a full analysis on their behavior. They cannot be treated as independent language models due to the fact that neural networks are only used to estimate the probabilities for *n*-grams ending with in-shortlist word, whereas conventional *n*-gram language models are still used to normalize these probabilities and provide the other ones.

The first contribution of this thesis is hence devoted to a new architecture , Structure OUtput Layer (SOUL) architecture, that makes them possible to be used with all possible *n*-grams. This modification is shown to bring significant improvements on large scale machine translation and automatic speech recognition tasks in different languages. Several refinements of the structure for this new kind of model are also proposed. The final version with a compact output hierarchical structure and with two hidden layers is demonstrated to not only drastically reduce the computational time but also yield additional gains in terms of both intrinsic measure (perplexity) and extrinsic measure (BLEU for SMT and WER for ASR).

Although the use of a continuous word space is a main idea, in practice, the quality of output word spaces is often overlooked. For this reason, several further analyses are performed in Chapter 4. It is shown that in their original architecture, there are actually two word spaces that encode two different (context and prediction) roles. They are similar but not identical. By drawing the neighborhoods of some random words, SOUL NNLMs are proved to be able to induce context word spaces, which encode some semantic and syntactic properties, in different languages. Their spaces are also evaluated on another task, a word relatedness measure, and are shown to achieve comparable results with state-of-the-art approaches. It means that the information taken from NNLMs can be also useful for other NLP tasks.

The advantage of using longer contexts of NNLMs is deeply studied in Chapter 5. Even though SOUL NNLMs can use longer context words than conventional *n*-gram back-off LMs, there exists also a limitation for them, at least in practice. In the machine translation evaluation of WMT 2011, history words further than the 9 previous words seem to have a very little impact on system performances. Max SOUL NNLMs, despite their degradation on the performance, have been shown to provide a straightforward way to measure the influence of context words. A new training scheme for recurrent neural network language models (RNNLMs) is also proposed in this chapter. Experiment results show that it is possible to train RNNLMs on a large scale machine translation task[1] to achieve comparable performances.

As finally shown, adopting SOUL architecture to *n*-gram translation models brings large improvements over state-of-the-art systems for large scale machine translation tasks adapted from IWSLT 2001 and WMT 2012 evaluations. It demonstrates that the continuous space based approach is generally an effective remedy to the data sparsity issue that should not be restricted to language modeling.

Future work

Even though using more than two hidden layers in SOUL NNLMs is not yet proved to be useful as seen in Section 3.3, following a deep learning approach with more complicated techniques such as those presented in (Hinton, Osindero, and Teh, 2006; Bengio, 2009) is a possible source of improvement. A motivation for this idea is that the deep learning approach has been shown to consistently achieve state-of-the-art results in many fields, notably in the domain of image processing (Hinton, Osindero, and Teh, 2006), of acoustic modeling (Seide, Li, and Yu, 2011; Mohamed, Dahl, and Hinton, 2012)...In general, it is necessary to find an effective way to deal with local optima of neural networks. Replacing neural networks with other types of models that could still make good use of continuous word space representation, is another interesting direction.

[1]The training data for language models is about 2.5 billion words

Avoiding the n-gram assumption is an interesting idea despite the fact that the RNNLMs did not outperform NNLMs of a large order as demonstrated in Chapter 5. Representing the whole context in a more sophisticated way similar to the work related to Structured Language Model in (Emami, 2006) needs to be examined. Another interesting way is presented in (Socher et al., 2011), where variable-sized phrases can be represented in a continuous space thanks to the introduction of the syntactic structure into the neural network architecture.

As NNLMs can be used to deal with many kind of objective function, another source of improvement is to find a way to train NNLMs discriminatively, i.e., by optimizing directly extrinsic objective functions for each task (BLEU for SMT, WER for ASR ...), following, e.g., the work of Xu et al. (2012).

NNLMs with SOUL structure can actually be trained on data of several billion words. However, pushing them to an extreme large scale order remains challenging. Finding solutions to solve this issue is therefore an important direction. It could be done by adopting data sampling techniques similar to the work of Bengio and Senecal (2008) or by using more advanced parallel programming techniques, e.g., using large clusters similar to the work in (Le et al., 2012b), using graphics cards as done in (Schwenk, Rousseau, and Attik, 2012).

It remains expensive to use NNLMs in applications such as ASR and SMT. Therefore, conventional back-off LMs are always required to efficiently build the search space that will then be rescored with NNLMs. To achieve state-of-the-art performances in a reasonable time, it is worth finding a tighter integration of continuous space neural network models into SMT and ASR systems, for example, the work of Lecorvé and Motlicek (2012). How to directly incorporate the continuous features induced from the word representations of NNLMs and the other discrete ones is therefore an interesting question. As a consequence, the behavior of continuous space based LMs compared to discrete based ones needs to be further analyzed as recently done in (Oparin et al., 2012).

As exploiting the knowledge encoded in continuous word spaces can potentially bring improvements as suggested by preliminary results on word relatedness task presented in Section 4.3, using similar structures for other NLP tasks is also a promising direction.

Appendix A
Abbreviation

- ATLAS: Automatically Tuned Linear Algebra Software
- ASR: Automatic Speech Recognition
- BLAS: Basic Linear Algebra Subprograms
- BLEU: Bilingual Evaluation Understudy
- BPTT: Back-Propagation Through Time
- CER: Character Error Rate
- DTLM: Decision Tree Language Model
- EM: Expectation-Maximization (algorithm)
- ESA: Explicit Semantic Analysis
- GALE: Global Autonomous Language Exploitation
- HCI: Human Computer Interface
- HLBL: Hierarchical Log Bi-Linear
- IE: Information Extraction
- IWSLT: International Workshop on Spoken Language Translation
- LBL: Log Bi-Linear
- LDA: Latent Dirichlet Allocation
- LIMSI: Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur (Computer Sciences Laboratory for Mechanics and Engineering Sciences)
- LM: Language Model
- LSA: Latent Semantic Analysis
- NIST: National Institute of Standards and Technology
- NLP: Natural Language Processing
- MAP: Maximum A Posteriori
- METEOR: Metric for Evaluation of Translation with Explicit ORdering
- MERT: Mininum Error Rate Training
- MIRA: Infused Relaxed Algorithm
- MLE: Maximum Likelihood Estimate
- NNLM: Neural Network Language Model
- MT: Machine Translation
- OOS: Out-Of-Shortlist
- PCA: Principal Component Analysis
- PLSA: Probabilistic Latent Semantic
- ppl: perplexity
- POS: Part-Of-Speech
- PRO: Pair Wise Ranking Optimization
- SE: Search Engine
- SGD: Stochastic Gradient Descend
- SMT: Statistical Machine Translation
- SRL: Semantic Role Labeling
- SLM: Structured Language Model
- SOUL: Structured OUtput Layer
- TER: Translation Error Rate
- TSA: Temporal Semantic Analysis
- TTS: Text to Speech
- RFLM: Random Forest Language Model
- RNNLM: Recurrent Neural Network Language Model
- WER: Word Error Rate
- WMT: Wordshop on Machine Translation

Appendix B

Word Space Examples for SOUL NNLMs

In this section, word space examples for SOUL NNLMs trained on data provided by WMT evaluation campaign 2012 are represented in the similar way as in Section 4.2. Word similarity is analyzed by finding the nearest neighbors according to the Euclidean distance in the projection space for random words selected from the 10,000 most frequent words.

Table B.1: *Examples of nearest neighbors according to the SOUL NNLM projection space for French.*

word	**nearest neighbors**
paroisse	pensionnat - paroissiaux - paroissiales - paroissial - évêché - archevêché - Paroisses - diocèse - hospice
biotechnologie	nanotechnologie - photonique - bioinformatique - protéomique - nanotechnologies - microélectronique - géomatique - biotechnologies - microfinance
64	63 - 66 - 55 - 62 - 51 - 60 - 52 - 57 - 53
Renault	VW - Michelin - Nissan - Mercedes-Benz - Safran - Faurecia - Renault-Nissan - Volkswagen - Thalès
ouvre	ouvrait - ouvrent - ouvrira - ouvrir - ouvrirait - ouvraient - ouvrant - ouvriront - ouvrit
occupé	occupées - occupa - occupée - occupés - occuperait - occuperont - bannie - occupons - assumées
espèces	Espèces - essences - sous-espèces - sous-populations - plantules - provenances - taxons - graminées - boutures
tuée	assassinée - assassinés - égorgé - assassinées - égorgés - kidnappé - incendiée - massacrés - suicidée
fibres	résines - cires - levures - teintures - filaments - Huiles - Fibres - gélules - caoutchoucs
victime	fuyard - malfaiteur - protagoniste - prisonnière - ravisseuse - séquelle - colocataire - coaccusé - affligée
uranium	plutonium - centrifugeuses - MOX - Yongbyon - atomiques - Natanz - radioactifs - fissile - thorium
évolution	décroissance - dynamisation - structuration - inflexion - atonie - fléchissement - infléchissement - hétérogénéité - réalignement
voulaient	voudraient - veulent - souhaitaient - voulions - voulait - veuillent - aimeraient - voulut - désiraient
Jeux	paralympiques - Paralympiques - J.O. - Olympiques - olympiques - Universiades - J0 - médaillées - anneaux
bienvenue	Bienvenue - bienvenu - bienvenus - bienvenues - félicitation - main-forte - liminaires - transposable - attitré
écrites	lues - transcrites - manuscrites - documentées - archivées - authentifiée - abrégés - authentifiées - manuscrite
Google	Microsoft - Amazon - MySpace - Motorola - Skype - eBay - YouTube - IBM - MSN
prévalence	Prévalence - étiologie - fécondité - pathogenèse - séropositivité - séroprévalence - Incidence - VHC - vaccinale
350	300 - 260 - 950 - 150 - 550 - 370 - 380 - 320 - 650

Table B.2: *Examples of nearest neighbors according to the SOUL NNLM projection space for German.*

word	nearest neighbors
Laptop	Notebook - Fotoapparat - Digitalkamera - Navigationsgeräte - Blackberry - Walkman - MP3-Player - Webcam - Anrufbeantworter
bürgerliche	Bürgerliche - sozialliberale - pro-europäische - proeuropäische - wirtschaftsliberale - sozial-liberale - bürgerlicher - christlich-demokratische - konservativ-liberale
New	Lynndie - Serious - Alianza - südöstlichsten - wiedervereinten - Biscayne - unwichtiges - Connaught - säumiger
stimmt	stimmen - stimmte - zutrifft - aufkommt - zusammenhängt - überrasche - bessert - misslingt - mutete
Kurve	Startposition - Vorderrad - Sohle - Mittellinie - Diskus - Piste - Drehzahl - Gaspedal - Boxengasse
vorgelegten	ausgearbeiteten - eingebrachten - vorgestellten - vorzulegenden - herausgegebenen - unterbreiteten - eingereichten - erörterten - umgesetzten
lauten	lauteten - ertönen - tönen - skandieren - zurufen - verstummten - zeitigen - welken - dumpfen
Carl	Corazza - Bertha - Othmar - Nathan - C.H. - Henrik - Jens-Peter - Jewgeni - H.W.
Ausnahmezustand	Kriegsrecht - Notstand - Nachrichtensperre - Belagerungszustand - Notstandsgesetze - Demonstrationsverbot - Kriegszustand - Ausgangssperren - Arrest
hab	hättest - hast - habe - werd - brauch - Habe - kriege - hattest - kriegst
schaden	schadeten - misstrauen - nützen - gehorchen - verderben - beschädigen - schädigen - entspringen - überfordern
gewannen	siegten - unterlagen - errangen - festigten - gewann - besiegten - vergaben - triumphierten - kassierten
Bruttoinlandsprodukt	Wirtschaftsleistung - Bruttoinlandsproduktes - Bruttosozialprodukt - Bruttoinlandsprodukts - Lohnstückkosten - Bruttoinlandprodukts - Konsumausgaben - Wirtschaftskraft - Volkseinkommen
Hindukusch	Afghanistan-Krieg - Nil - Anbar - Gaddafi-Clan - Georgien-Konflikt - Ostkongo - Osttürkei - Golan - Bundesheer
gestand	gesteht - beteuerte - räumten - zugab - gestehe - leugnete - mutmaßte - stöhnte - schwor
Wahnsinn	Irrsinn - Schwachsinn - Größenwahn - Jammer - Jugendwahn - Selbstbetrug - ok - Donnerwetter - Blödsinn
Cockpit	Bodenpersonal - Eurostar - Cockpits - Flugplan - Condor - Schaltung - Germanwings - Motorraum - Fluglotse
blauen	blaue - gelben - gestreiften - rosafarbenen - blütenweißen - cremefarbenen - blauer - violetten - orangefarbenen
Gefühle	Empfindungen - Regungen - Leidenschaften - Phantasien - Fantasien - Emotionen - Instinkte - Schuldgefühle - Charaktereigenschaften
deine	Deine - deiner - Eure - eure - deinen - Deiner - euren - Deinen - eurer

Table B.3: *Examples of nearest neighbors according to the SOUL NNLM projection space for Spanish.*

word	nearest neighbors
afiliados	agremiados - afiliadas - adheridos - afiliado - afiliada - adherentes - adscritas - empadronados - cotizantes
tristeza	angustia - desesperanza - desánimo - desilusión - amargura - asombro - estupor - resignación - incredulidad
debatiendo	discutiendo - examinando - debata - discutimos - debatimos - debatamos - debatiremos - debatió - debaten
afirmado	subrayado - recalcado - aseverado - remarcado - opinado - argumentado - comentado - puntualizado - apostillado
bienes	patrimonios - inmuebles - dineros - equipamientos - implementos - tesoros - haberes - electrodomésticos - enseres
telefónicas	telefónicos - pericias - hospitalarias - escritorios - juradas - billetera - faxes - e-mails - golpizas
celebraron	celebraban - celebraba - festejaron - celebrando - conmemoraron - celebran - celebraran - conmemoran - celebrara
cómplices	cómplice - victimarios - perpetradores - acompañantes - ejecutores - homicidas - acompañante - subalternos - victimas
cenizas	ceniza - Cenizas - deslave - sunami - brisa - nubosas - nevada - vórtice - eruptiva
así	asi - preliminarmente - asÍ - paulatinamente - instantáneamente - seguidamente - Asi - sabiamente - enseguida
efectuó	efectuaron - efectuaba - efectuaban - efectúa - realizaría - realizara - efectuara - realizo - realizaran
naturaleza	moralidad - luminosidad - índole - abstracción - singularidad - textura - bondad - tipología - homogeneidad
explican	comentan - subrayaron - subrayan - relatan - recalcan - asevera - recalcaron - aclararon - resaltaron
vinculación	nexo - vinculaciones - interlocución - interrelación - connivencia - interacción - compatibilidad - afinidad - colusión
directo	indirecto - expresivo - humorístico - separadamente - tiránico - acrobático - vitalicio - reposado - eufórica
prestan	prestarán - presten - prestaba - prestaban - prestarles - prestará - prestaran - prestarle - prestaremos
Prado	Valme - Casal - Barranco - Atienza - Cabezas - Alcázar - Martirio - Eroski - Bermejales
Líbano	Argelia - Tíbet - Zimbabue - Jordania - Chechenia - Golán - Darfur - palestinas - OLP
serias	gravísimas - tremendas - severas - consiguientes - obvias - excesivas - auténticas - significativas - drásticas
autorización	autorizaciones - anuencia - acreditación - requerimiento - notificación - salvoconducto - citación - papeleo - validación

Appendix C

Derivatives of the SOUL objective function

Remind that as described in Section 2.1, w_n is used to denote the word to be predicted. The sequence $x_0^U(w_n) = x_0, \ldots, x_U$ is used to encode the path for word w_n in the clustering tree. x_0 is the root of the tree, x_u with $u = 1, \ldots, U-1$ are the class assigned to w_n and x_U is the leaf associated with w_n. The probability of w_n given its history w_1^{n-1} can then be computed as:

$$P(w_n|w_1^{n-1}) = \prod_{u=1}^{U} P(x_u|w_1^{n-1}, x_{u-1}) \tag{C.1}$$

We introduce here $\zeta_1^U(w_n) = \zeta_1\zeta_2\ldots\zeta_U$ where each component ζ_u is used to denote the index of x_u with respect to its father x_{u-1}. The first part of the objective function in Equation (1.80) for this example becomes:

$$\log P(w_n|w_1^{n-1}) = \sum_{u=1}^{U} \log P(x_u|w_1^{n-1}, x_{u-1}) \tag{C.2}$$

$$= \sum_{u=1}^{U} \log p_{\zeta_u}^{x_{u-1}} \tag{C.3}$$

where $\mathbf{p}^{\mathbf{x_{u-1}}}$ stands for the output value of the softmax layer associated to node x_{u-1}, $\mathbf{o}^{\mathbf{x_{u-1}}}$ is its associated value before normalization.

At the output part, derivatives for each u are computed as follows:

$$\Delta o_i^{x_{u-1}} = \begin{cases} 1 - p_i^{x_{u-1}} & \text{if } i = \zeta_u \\ -p_i^{x_{u-1}} & \text{otherwise} \end{cases} \tag{C.4}$$

Then, the derivatives for a softmax layer associated with node x_{u-1} are computed as follows:

$$\Delta \mathbf{b}^{\mathbf{x_{u-1}}} = \Delta \mathbf{o}^{\mathbf{x_{u-1}}}, \tag{C.5}$$

$$\Delta \mathbf{W}^{\mathbf{x_{u-1}}} = \Delta \mathbf{o}^{\mathbf{x_{u-1}}} \mathbf{h}^T, \tag{C.6}$$

(C.7)

where $\mathbf{h}$ is the last hidden layer, which directly connects to the output part, $\mathbf{b}^{\mathbf{x_{u-1}}}$ and $\mathbf{W}^{\mathbf{x_{u-1}}}$ are the parameters of the softmax layer.

The derivatives for $\mathbf{h}$:

$$\Delta \mathbf{h} = \sum_{u=1}^{U} \mathbf{W}^{\mathbf{x_{u-1}}T} \Delta \mathbf{o}^{\mathbf{x_{u-1}}} \tag{C.8}$$

After that, the derivatives from the other hidden layers back to the input layer are computed using the same equations as with feed-forward NNLMs (see Section 1.4.2.2).

Appendix D

Implementation Issues

For reference, all training times reported in this thesis are on a *8 x Intel(R) Xeon(R) 1.86GHz* server. Our implementation is in C++ to benefit from the class concept. The preliminary idea for the source code architecture for neural network (definitions of tensors, modules . . .) is from *Torch5*[1] toolkit. Our toolkit is also influenced by other toolkits for neural network language models: *CSLM*[2] and *RNNLM*[3].

Real number precision Real number can be represented by 32bits (float) or 64bits (double). Using float is faster and doesn't degrade the system performance.

Bunch mode This method aims at propagating several examples at once through the network. To implement that, there is a popular mathematical library Basic Linear Algebra Subprograms (BLAS) which have optimized versions for several CPU architectures. For example, *MKL*[4] for Intel processors and *ACML*[5] for AMD processors. There exists also Automatically Tuned Linear Algebra Software (ATLAS)[6] that can be used to complie BLAS library in a nearly optimal adaptation fasion for each architecture. In our toolkit, we use the C interface to the BLAS[7].

Border effect We need to add $n-1$ <s> at the beginning of the sentence. For example, with the sentence *"Let music be the food of love"*, 4-gram NNLMs estime the probabilities: $P(\text{Let}|\text{<s> <s> <s>})$, $P(\text{music}|\text{<s> <s> Let})$, $P(\text{be}|\text{<s> Let music})$, $P(\text{the}|\text{ Let music be})$ . . .

Order When possible, the order of models, n, should be at least 6, 8 is adviced and 10 is in most cases an upper-bound value.

Data resampling We carry out separately data resampling from the training procedure. We randomly select n-grams from text files with respect to resampling rates, charging all in memory and then shuffling them by exchanging the positions of pairs of n-grams for a large number of times. The resulting n-grams are saved to disk in binary format. It will then be read one by one by the training procedure.

[1] http://torch5.sourceforge.net/index.html
[2] http://www-lium.univ-lemans.fr/cslm/
[3] http://www.fit.vutbr.cz/~imikolov/rnnlm/
[4] http://software.intel.com/en-us/articles/intel-mkl/
[5] http://developer.amd.com/libraries/acml/pages/default.aspx
[6] http://math-atlas.sourceforge.net
[7] http://www.netlib.org/blas/

Resampling rates If we have several data sets from different sources, resampling rates should be chosen to satisfy that, at each epoch, there are more n-grams from in-domain data than from out-of-domain data. Conditional back-off n-gram language models can be used to roughly determine which is in-domain or out-of-domain. A particular example is for WMT evaluations where we have an in-domain data set (News), other sets can be considered as out-of-domain. In this case, for each epoch, there should be 75% in-domain n-grams after resampling.

With the same total number of examples used in the training proceduce, larger resampling rates with a smaller number of epochs often have a better performance. For example, training models using a rate of 15% with 1 epoch is better than 5% with 3 epochs.

Learning rate For the learning rate, the second technique presented at the end of Section 1.4.2.2 is easier to use as it has less number of hyper-parameters (the learning rate decay is not required). The learning rate depends on other parameters: the dimension of projection space, the number, the size and the activation type of hidden layers H, the number of seen examples for each epoch. When using $m = 500$, $H = 1000 - 500$ with sigmoid, a good learning rate is $\xi_0 = 1 \times 10^{-2}$.

Weight decay $\mu' = 3 \times 10^{-5}$ is a good value.

Block size Block size should be set to 128. Using greater value doesn't make systems run faster.

Context grouping To compute probabilities, for SOUL NNLMs, context grouping is also useful as all computation from the input to the main softmax layer is shared by n-grams which have the same context.

Parameter initialization The initial values should be drawn from the uniform distribution in $[-10^{-2}, 10^{-2}]$.

Input part The projection dimension, M, should be set to 500. A larger value seems not to be very useful.

Hidden part There should be two sigmoid hidden layers of 1000 and 500 units ($H = 1000 - 500$). Using more than two hidden layers without pre-training using deep learning techniques is not helpful.

Output part The size of the shortlist and the number of top classes for OOS words should be 2000 and 2000 respectively. The depth of the output structure is 3: top classes with more than $s = 1000$ words should be divided into $\sqrt{s+1}$ subclasses.

Example Here is our configuration for the WMT 2012 evaluation:

10-gram, $m = 500$, $H = 1000, 500$, sigmoid, 2000 shortlist words and 2000 top classes.

Resampling rates are chosen to guarantee that models see 75% n-grams of in-domain data (News) and 25% n-grams of the other data. After resampling, there are 140 million examples for each epoch of Step 1, Step 3 and 50 million examples for Step 4. Number of epochs is $2, 2, 10$ for Step $1, 3, 4$ respectively.

On a *8 x Intel(R) Xeon(R) 1.86GHz* server, it takes about 10 days for training. The resulting models outperform conventional back-off n-gram language models in all languages (English, French, German).

Appendix E

Publications by the Author

- 2013
  - **Hai-Son Le**, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. Structured Output Layer Neural Network Language Models for Speech Recognition. In *IEEE Transactions on Audio, Speech and Language Processing* VOL. 21, NO. 1, January 2013.
- 2012
  - **Hai-Son Le**, Alexandre Allauzen, and François Yvon. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL)*, Montréal, Canada, 2012.
  - **Hai-Son Le**, Alexandre Allauzen, and François Yvon. Measuring the influence of long range dependencies with neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, Montréal, Canada, 2012.
  - **Hai-Son Le**, Thomas Lavergne, Alexandre Allauzen, Marianna Apidianaki, Li Gong, Aurélien Max, Artem Sokolov, Guillaume Wisniewski, and François Yvon. LIMSI @ WMT12. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, Montréal, Canada, 2012.
- 2011
  - Artem Solokov, Tanguy Urvoy, and **Hai-Son Le**. Low-dimensional feature learning with kernel construction. In *Proceedings of the NIPS 2011 Workshop on Deep Learning and Unsupervised Feature Learning*, Granada, Spain, 2011.
  - Thomas Lavergne, Alexandre Allauzen, **Hai-Son Le**, and François Yvon. LIMSI's experiments in domain adaptation for IWSLT11. In *Proceedings of the 8th International Workshop on Spoken Language Translation (IWSLT)*, San Francisco, CA, 2011.
  - Alexandre Allauzen, Gilles Adda, Hélène Bonneau-Maynard, Josep M. Crego, **Hai-Son Le**, Aurélien Max, Adrien Lardilleux, Thomas Lavergne, Artem Sokolov, Guillaume Wisniewski, and François Yvon. LIMSI @ WMT11. In *Proceedings of the 6th Workshop on Statistical Machine Translation (WMT)*, Edinburgh, Scotland, 2011.

  - **Hai-Son Le**, Ilya Oparin, Abdel. Messaoudi, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. Large vocabulary SOUL neural network language models. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association (Interspeech)*, Florence, Italy, 2011.
  - **Hai-Son Le**, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. Structured output layer neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech, 2011.

- 2010
  - Alexandre Allauzen, Josep M. Crego, Ilknur Durgar El-Kahlout, **Hai-Son Le**, Guillaume Wisniewski, and François Yvon. LIMSI @ IWSLT 2010. In *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*, Paris, France, 2010.
  - **Hai-Son Le**, Alexandre Allauzen, Guillaume Wisniewski, and François Yvon. Training continuous space language models: Some practical issues. In *Proceedings of the 2010 Conference on Empirical Methods on Natural Language Processing (EMNLP)*, Cambridge, MA, 2010.

Bibliography

Allauzen, Alexandre, Hélène Bonneau-Maynard, Hai-Son Le, Aurélien Max, Guillaume Wisniewski, François Yvon, Gilles Adda, Josep Maria Crego, Adrien Lardilleux, Thomas Lavergne, and Artem Sokolov (2011). "LIMSI @ WMT11". In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland: Association for Computational Linguistics, pp. 309–315. URL: http://www.aclweb.org/anthology/W11-2135 (cit. on pp. 62, 69, 124).

Andres-Ferrer, J., M. Sundermeyer, and H. Ney (2012). "Conditional leaving-one-out and cross-validation for discount estimation in Kneser-Ney-like extensions". In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 5013 –5016. DOI: 10.1109/ICASSP.2012.6289046 (cit. on p. 14).

Arisoy, Ebru, Tara N. Sainath, Brian Kingsbury, and Bhuvana Ramabhadran (2012). "Deep Neural Network Language Models". In: *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. Montréal, Canada: Association for Computational Linguistics, pp. 20–28. URL: http://www.aclweb.org/anthology/W12-2703 (cit. on p. 74).

Bahl, Lalit R., Peter F. Brown, Peter V. de, and Robert L. Mercer (1989). "A tree-based statistical language model for natural language speech recognition". In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 37.7, pp. 1001 –1008. ISSN: 0096-3518. DOI: 10.1109/29.32278 (cit. on p. 23).

Bellegarda, J.R. (1998). "A multispan language modeling framework for large vocabulary speech recognition". In: *Speech and Audio Processing, IEEE Transactions on* 6.5, pp. 456 –467. ISSN: 1063-6676. DOI: 10.1109/89.709671 (cit. on pp. 15, 21).

Bengio, Y. and J.-S. Senecal (2008). "Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model". In: *Neural Networks, IEEE Transactions on* 19.4, pp. 713 –722. ISSN: 1045-9227. DOI: 10.1109/TNN.2007.912312 (cit. on p. 131).

Bengio, Yoshua (01/2009). "Learning Deep Architectures for AI". In: *Found. Trends Mach. Learn.* 2.1, pp. 1–127. ISSN: 1935-8237. DOI: 10.1561/2200000006. URL: http://dx.doi.org/10.1561/2200000006 (cit. on pp. 73, 79, 82, 130).

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent (2000). "A Neural Probabilistic Language Model". In: *NIPS*. Ed. by Todd K. Leen, Thomas G. Dietterich, and Volker Tresp. MIT Press, pp. 932–938. URL: http://www.iro.umontreal.ca/~lisa/pointeurs/nips00_lm.ps (cit. on pp. xviii, 29–31, 43, 83, 109).

Bengio, Yoshua and Yann LeCun (2007). "Scaling Learning Algorithms towards AI". In: *Large Scale Kernel Machines*. Ed. by Léon Bottou, Olivier Chapelle, D. DeCoste, and J. Weston. MIT Press. URL: http://www.iro.umontreal.ca/~lisa/pointeurs/bengio+lecun_chapter2007.pdf (cit. on p. 73).

Bilmes, J., K. Asanovic, Chee-Whye Chin, and J. Demmel (1997). "Using PHiPAC to speed error back-propagation learning". In: *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*. Vol. 5, 4153 –4156 vol.5. DOI: 10.1109/ICASSP.1997.604861 (cit. on p. 15).

Bilmes, Jeff A. and Katrin Kirchhoff (2003). "Factored language models and generalized parallel backoff". In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers - Volume 2*. NAACL-Short '03. Edmonton, Canada: Association for Computational Linguistics, pp. 4–6. DOI: 10.3115/1073483.1073485. URL: http://dx.doi.org/10.3115/1073483.1073485 (cit. on pp. 15, 117).

Bimbot, Frédéric, Marc El-Bèze, Stéphane Igounet, Michèle Jardino, Kamel Smaili, and Imed Zitouni (2001). "An alternative scheme for perplexity estimation and its assessment for the evaluation of language models". In: *Computer Speech & Language* 15.1, pp. 1 –13. ISSN: 0885-2308. DOI: 10.

1006/csla.2000.0150. URL: http://www.sciencedirect.com/science/article/pii/S0885230800901505 (cit. on p. 7).

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (03/2003). "Latent dirichlet allocation". In: *J. Mach. Learn. Res.* 3, pp. 993–1022. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=944919.944937 (cit. on p. 22).

Brants, Thorsten, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean (2007). "Large Language Models in Machine Translation". In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 858–867. URL: http://www.aclweb.org/anthology/D/D07/D07-1090 (cit. on pp. 7, 14, 101).

Breiman, Leo, J. H. Friedman, R. A. Olshen, and C. J. Stone (1984). *Classification and Regression Trees*. Chapman and Hall (cit. on p. 23).

Brown, Peter F., Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai (12/1992). "Class-based n-gram models of natural language". In: *Comput. Linguist.* 18.4, pp. 467–479. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=176313.176316 (cit. on pp. xviii, 15, 28, 60).

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer (06/1993). "The mathematics of statistical machine translation: parameter estimation". In: *Comput. Linguist.* 19.2, pp. 263–311. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=972470.972474 (cit. on pp. 118, 121).

Callison-Burch, Chris, Miles Osborne, and Philipp Koehn (2006). "Re-evaluating the Role of BLEU in Machine Translation Research". In: *In EACL*, pp. 249–256 (cit. on p. 9).

Callison-Burch, Chris, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia (2012). "Findings of the 2012 Workshop on Statistical Machine Translation". In: *Proceedings of the Seventh Workshop on Statistical Machine Translation*. Montréal, Canada: Association for Computational Linguistics, pp. 10–51. URL: http://www.aclweb.org/anthology/W12-3102 (cit. on p. 127).

Casacuberta, Francisco and Enrique Vidal (06/2004). "Machine Translation with Inferred Stochastic Finite-State Transducers". In: *Comput. Linguist.* 30.2, pp. 205–225. ISSN: 0891-2017. DOI: 10.1162/089120104323093294. URL: http://dx.doi.org/10.1162/089120104323093294 (cit. on p. 119).

Chelba, Ciprian and Frederick Jelinek (2000). "Structured language modeling". In: *Computer Speech & Language* 14.4, pp. 283 –332. ISSN: 0885-2308. DOI: 10.1006/csla.2000.0147. URL: http://www.sciencedirect.com/science/article/pii/S0885230800901475 (cit. on pp. xviii, 17, 18, 101).

Chen, S.F. and R. Rosenfeld (2000). "A survey of smoothing techniques for ME models". In: *Speech and Audio Processing, IEEE Transactions on* 8.1, pp. 37 –50. ISSN: 1063-6676. DOI: 10.1109/89.817452 (cit. on p. 26).

Chen, S.F., K. Seymore, and R. Rosenfeld (1998). "Topic adaptation for language modeling using unnormalized exponential models". In: *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. Vol. 2, 681 –684 vol.2. DOI: 10.1109/ICASSP.1998.675356 (cit. on p. 22).

Chen, Stanley F. (2009a). "Performance prediction for exponential language models". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. NAACL '09. Boulder, Colorado: Association for Computational Linguistics, pp. 450–458. ISBN: 978-1-932432-41-1. URL: http://dl.acm.org/citation.cfm?id=1620754.1620820 (cit. on p. 27).

Chen, Stanley F. (2009b). "Shrinking exponential language models". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. NAACL '09. Boulder, Colorado: Association for Computational Linguistics, pp. 468–476.

ISBN: 978-1-932432-41-1. URL: http://dl.acm.org/citation.cfm?id=1620754.1620822 (cit. on pp. 15, 27, 28).

Chen, Stanley F. and Stephen M. Chu (2010). "Enhanced word classing for model M". In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, pp. 1037–1040. URL: http://www.isca-speech.org/archive/interspeech_2010/i10_1037.html (cit. on pp. 28, 58).

Chen, Stanley F. and Joshua Goodman (1998). "An empirical study of smoothing techniques for language modeling". In: *Technical Report TR-10-98*. Cambridge, Massachusetts, USA: Computer Science Group, Harvard University (cit. on pp. xvii, 11, 13, 69, 125).

Chen, Stanley F., Lidia Mangu, Bhuvana Ramabhadran, Ruhi Sarikaya, and Abhinav Sethy (2009). "Scaling shrinkage-based language models". In: *IEEE Workshop on Automatic Speech Recognition and Understanding*. DOI: 10.1109/ASRU.2009.5373380 (cit. on p. 28).

Chen, Stanley, Douglas Beeferman, and Ronald Rosenfeld (1998). *Evaluation Metrics For Language Models* (cit. on p. 7).

Cherry, Colin and George Foster (2012). "Batch Tuning Strategies for Statistical Machine Translation". In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montréal, Canada: Association for Computational Linguistics, pp. 427–436. URL: http://www.aclweb.org/anthology/N12-1047 (cit. on p. 6).

Chiang, David, Kevin Knight, and Wei Wang (2009). "11,001 new features for statistical machine translation". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. NAACL '09. Boulder, Colorado: Association for Computational Linguistics, pp. 218–226. ISBN: 978-1-932432-41-1. URL: http://dl.acm.org/citation.cfm?id=1620754.1620786 (cit. on p. 6).

Chiang, David, Yuval Marton, and Philip Resnik (2008). "Online large-margin training of syntactic and structural translation features". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '08. Honolulu, Hawaii: Association for Computational Linguistics, pp. 224–233. URL: http://dl.acm.org/citation.cfm?id=1613715.1613747 (cit. on p. 6).

Chien, Jen-Tzung and Chuang-Hua Chueh (03/2011). "Dirichlet Class Language Models for Speech Recognition". In: *Trans. Audio, Speech and Lang. Proc.* 19.3, pp. 482–495. ISSN: 1558-7916. DOI: 10.1109/TASL.2010.2050717. URL: http://dx.doi.org/10.1109/TASL.2010.2050717 (cit. on p. 22).

Collins, Michael, Brian Roark, and Murat Saraclar (2005). "Discriminative Syntactic Language Modeling for Speech Recognition". In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 507–514. DOI: 10.3115/1219840.1219903. URL: http://www.aclweb.org/anthology/P05-1063 (cit. on p. 101).

Collobert, Ronan and Jason Weston (2008). "A unified architecture for natural language processing: deep neural networks with multitask learning". In: *Proc. of ICML'08*. Helsinki, Finland: ACM, pp. 160–167. ISBN: 978-1-60558-205-4. DOI: http://doi.acm.org/10.1145/1390156.1390177 (cit. on pp. 30, 40, 83, 87, 97, 103, 104).

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa (11/2011). "Natural Language Processing (Almost) from Scratch". In: *J. Mach. Learn. Res.* 999888, pp. 2493–2537. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2078183.2078186 (cit. on pp. 40, 41).

Crego, Josep M., François Yvon, and José B. Mariño (2011). "N-code: an open-source Bilingual N-gram SMT Toolkit". In: *Prague Bulletin of Mathematical Linguistics* 96, pp. 49–58. DOI: 10.2478/v10108-011-0010-5. URL: http://dx.doi.org/10.2478/v10108-011-0010-5 (cit. on pp. 69, 124).

Crego, Josep Maria and José B. Mariño (09/2006). "Improving statistical MT by coupling reordering and decoding". In: *Machine Translation* 20.3, pp. 199–215. ISSN: 0922-6567. DOI: 10.1007/s10590-007-9024-z. URL: http://dx.doi.org/10.1007/s10590-007-9024-z (cit. on pp. 119, 124).

Crego, Josep and François Yvon (2010). "Factored bilingual n-gram language models for statistical machine translation". In: *Machine Translation* 24 (2). 10.1007/s10590-010-9082-5, pp. 159–175. ISSN: 0922-6567. URL: http://dx.doi.org/10.1007/s10590-010-9082-5 (cit. on p. 117).

Dagan, Ido, Lillian Lee, and Fernando C. N. Pereira (02/1999). "Similarity-Based Models of Word Cooccurrence Probabilities". In: *Mach. Learn.* 34.1-3, pp. 43–69. ISSN: 0885-6125. DOI: 10.1023/A:1007537716579. URL: http://dx.doi.org/10.1023/A:1007537716579 (cit. on pp. 18, 19).

Dagan, Ido, Fernando Pereira, and Lillian Lee (1994). "Similarity-based estimation of word cooccurrence probabilities". In: *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. ACL '94. Las Cruces, New Mexico: Association for Computational Linguistics, pp. 272–278. DOI: 10.3115/981732.981770. URL: http://dx.doi.org/10.3115/981732.981770 (cit. on p. 18).

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman (1990). "Indexing by Latent Semantic Analysis". In: *JASIS* 41.6, pp. 391–407 (cit. on pp. 21, 118).

Diehl, Frank, Mark J. F. Gales, Marcus Tomalin, and Philip C. Woodland (2009). "Morphological analysis and decomposition for Arabic speech-to-text systems". In: *Proceedings of the 10th Annual Conference of the International Speech Communication Association (INTERSPEECH 2009)*, pp. 2675–2678. URL: http://www.isca-speech.org/archive/interspeech_2009/i09_2675.html (cit. on p. 64).

Elman, Jeffrey L. (1990). "Finding Structure in Time". In: *Cognitive Science* 14.2, pp. 179–211. ISSN: 1551-6709. DOI: 10.1207/s15516709cog1402_1. URL: http://dx.doi.org/10.1207/s15516709cog1402_1 (cit. on pp. 30, 37).

Emami, A. and S.F. Chen (2011). "Multi-class Model M". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pp. 5516 –5519. DOI: 10.1109/ICASSP.2011.5947608 (cit. on p. 58).

Emami, A. and F. Jelinek (2005). "Random Clusterings for Language Modeling". In: *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*. Vol. 1, pp. 581 –584. DOI: 10.1109/ICASSP.2005.1415180 (cit. on pp. 15, 16).

Emami, A. and L. Mangu (2007). "Empirical study of neural network language models for Arabic speech recognition". In: *Automatic Speech Recognition Understanding, 2007. ASRU. IEEE Workshop on*, pp. 147 –152. DOI: 10.1109/ASRU.2007.4430100 (cit. on pp. xix, 18, 30).

Emami, Ahmad (2006). "A Neural Syntactic Language Model". PhD thesis. Baltimore, MD, USA: Johns Hopkins University (cit. on pp. 30, 55, 57, 83, 131).

Emami, Ahmad, Imed Zitouni, and Lidia Mangu (2008). "Rich morphology based n-gram language models for Arabic". In: *Proceedings of the 9th Annual Conference of the International Speech Communication Association (INTERSPEECH 2008)*, pp. 829–832. URL: http://www.isca-speech.org/archive/interspeech_2008/i08_0829.html (cit. on p. 101).

Filimonov, Denis and Mary Harper (2009). "A Joint Language Model With Fine-grain Syntactic Tags". In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, pp. 1114–1123. URL: http://www.aclweb.org/anthology/D/D09/D09-1116 (cit. on p. 17).

Finkelstein, Lev, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin (2001). "Placing search in context: the concept revisited". In: *Proceedings of the 10th international conference on World Wide Web*. WWW '01. Hong Kong, Hong Kong: ACM, pp. 406–414. ISBN: 1-58113-348-0. DOI: 10.1145/371920.372094. URL: http://doi.acm.org/10.1145/371920.372094 (cit. on p. 95).

Foster, George, Roland Kuhn, and Howard Johnson (2006). "Phrasetable Smoothing for Statistical Machine Translation". In: *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Sydney, Australia: Association for Computational Linguistics, pp. 53–61. URL: http://www.aclweb.org/anthology/W/W06/W06-1607 (cit. on p. 117).

Gabrilovich, Evgeniy and Shaul Markovitch (2007). “Computing semantic relatedness using Wikipedia-based explicit semantic analysis”. In: *Proceedings of the 20th international joint conference on Artifical intelligence*. IJCAI'07. Hyderabad, India: Morgan Kaufmann Publishers Inc., pp. 1606–1611. URL: http://dl.acm.org/citation.cfm?id=1625275.1625535 (cit. on pp. 95, 97).

Gao, Qin and Stephan Vogel (2008). “Parallel implementations of word alignment tool”. In: *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*. SETQA-NLP '08. Columbus, Ohio: Association for Computational Linguistics, pp. 49–57. ISBN: 978-1-932432-10-7. URL: http://dl.acm.org/citation.cfm?id=1622110.1622119 (cit. on p. 124).

Gildea, Daniel and Thomas Hofmann (1999). “Topic-Based Language Models Using EM”. In: *Proceedings of the 6th European Conference on Speech Communication and Technology (EUROSPEECH 1999)*. Budapest, Hungary, pp. 2167–2170 (cit. on pp. 15, 20, 21).

Gillot, Christian and Christophe Cerisara (08/2011). “Similarity language model”. In: *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011)*. Florence, Italie, p. 4. URL: http://www.isca-speech.org/archive/interspeech_2011/i11_1457.html (cit. on p. 19).

Goodman, J. (2001a). “Classes for fast maximum entropy training”. In: *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*. Vol. 1, 561 –564 vol.1. DOI: 10.1109/ICASSP.2001.940893 (cit. on pp. 26, 57).

Goodman, Joshua T. (2001b). “A bit of progress in language modeling”. In: *Computer Speech & Language* 15.4, pp. 403 –434. ISSN: 0885-2308. DOI: 10.1006/csla.2001.0174. URL: http://www.sciencedirect.com/science/article/pii/S0885230801901743 (cit. on pp. 15, 16).

Gosme, Julien and Yves Lepage (2011). “Structure des trigrammes inconnus et lissage par analogie”. In: *18e TALN*. Montpellier, France (cit. on p. 19).

Griffiths, Thomas L. and Mark Steyvers (2004). “Finding Scientific Topics”. In: *PNAS* 101.suppl. 1, pp. 5228–5235 (cit. on p. 22).

Hinton, Geoffrey E. (1986). “Learning distributed representations of concepts”. In: *Proceedings of the Eighth Annual Conference of the Cognitive Siience Society*, pp. 1–12 (cit. on p. 30).

Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh (07/2006). “A fast learning algorithm for deep belief nets”. In: *Neural Comput.* 18.7, pp. 1527–1554. ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527. URL: http://dx.doi.org/10.1162/neco.2006.18.7.1527 (cit. on pp. 73, 82, 99, 130).

Hofmann, Thomas (1999). “Probabilistic latent semantic analysis”. In: *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. UAI'99. Stockholm, Sweden: Morgan Kaufmann Publishers Inc., pp. 289–296. ISBN: 1-55860-614-9. URL: http://dl.acm.org/citation.cfm?id=2073796.2073829 (cit. on p. 21).

Hopkins, Mark and Jonathan May (2011). “Tuning as Ranking”. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, pp. 1352–1362. URL: http://www.aclweb.org/anthology/D11-1125 (cit. on p. 6).

Huang, Eric, Richard Socher, Christopher Manning, and Andrew Ng (2012). “Improving Word Representations via Global Context and Multiple Word Prototypes”. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea: Association for Computational Linguistics, pp. 873–882. URL: http://www.aclweb.org/anthology/P12-1092 (cit. on pp. 95–97).

Jelinek, Frederick (1990). “Readings in speech recognition”. In: ed. by Alex Waibel and Kai-Fu Lee. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Chap. Self-organized language modeling for speech recognition, pp. 450–506. ISBN: 1-55860-124-4. URL: http://dl.acm.org/citation.cfm?id=108235.108270 (cit. on p. xviii).

Jelinek, Frederick (1995). “Closing remarks”. In: *The 1995 language modeling summer workshop at Johns Hopkins University* (cit. on p. 51).

Khudanpur, Sanjeev and Jun Wu (2000). “Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling”. In: *Computer Speech & Language* 14.4, pp. 355 –372. ISSN: 0885-2308. DOI: 10.1006/csla.2000.0149. URL: http://www.sciencedirect.com/science/article/pii/S0885230800901499 (cit. on p. 18).

Kneser, R. and H. Ney (1995). “Improved backing-off for M-gram language modeling”. In: *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. Vol. 1, 181 –184 vol.1. DOI: 10.1109/ICASSP.1995.479394 (cit. on pp. xvii, 11, 12, 69, 125).

Kneser, Reinhard and Hermann Ney (1993). “Improved clustering techniques for class-based statistical language modeling”. In: *In Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)* (cit. on p. 16).

Koehn, Philipp and Hieu Hoang (2007). “Factored Translation Models”. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 868–876. URL: http://www.aclweb.org/anthology/D/D07/D07-1091 (cit. on p. 117).

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst (2007). “Moses: open source toolkit for statistical machine translation”. In: *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. ACL ’07. Prague, Czech Republic: Association for Computational Linguistics, pp. 177–180. URL: http://dl.acm.org/citation.cfm?id=1557769.1557821 (cit. on pp. 96, 119, 127).

Kuo, H.-K.J., L. Mangu, A. Emami, I. Zitouni, and Young-Suk Lee (2009). “Syntactic features for Arabic speech recognition”. In: *Automatic Speech Recognition Understanding, 2009. ASRU 2009. IEEE Workshop on*, pp. 327 –332. DOI: 10.1109/ASRU.2009.5373470 (cit. on p. 18).

Kuo, H.-K.J., L. Mangu, A. Emami, and I. Zitouni (2010). “Morphological and syntactic features for Arabic speech recognition”. In: *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 5190 –5193. DOI: 10.1109/ICASSP.2010.5495010 (cit. on pp. xix, 30).

Lamel, Lori, Abdelkhalek Messaoudi, and Jean-Luc Gauvain (2008). “Investigating morphological decomposition for transcription of Arabic broadcast news and broadcast conversation data”. In: *Proceedings of the 9th Annual Conference of the International Speech Communication Association (INTERSPEECH 2008)*, pp. 1429–1432. URL: http://www.isca-speech.org/archive/interspeech_2008/i08_1429.html (cit. on p. 64).

Lamel, Lori, Abdelkhalek Messaoudi, and Jean-Luc Gauvain (12/2009). “Automatic Speech-to-Text Transcription in Arabic”. In: 8.4, 18:1–18:18. ISSN: 1530-0226. DOI: 10.1145/1644879.1644885. URL: http://doi.acm.org/10.1145/1644879.1644885 (cit. on pp. 62, 64).

Lau, Raymond, Ronald Rosenfeld, and Salim Roukos (1993). “Adaptive language modeling using the maximum entropy principle”. In: *Proceedings of the workshop on Human Language Technology*. HLT ’93. Princeton, New Jersey: Association for Computational Linguistics, pp. 108–113. ISBN: 1-55860-324-7. DOI: 10.3115/1075671.1075695. URL: http://dx.doi.org/10.3115/1075671.1075695 (cit. on pp. 15, 25).

Lavergne, Thomas, Alexandre Allauzen, Hai-Son Le, and François Yvon (2011). “LIMSI’s experiments in domain adaptation for IWSLT11”. In: *Proceedings of the 8th International Workshop on Spoken Language Translation (IWSLT)*. San Francisco, CA (cit. on p. 125).

Le, Hai-Son, Alexandre Allauzen, and François Yvon (2012a). “Continuous Space Translation Models with Neural Networks”. In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montréal, Canada: Association for Computational Linguistics, pp. 39–48. URL: http://www.aclweb.org/anthology/N12-1005 (cit. on p. 118).

Le, Hai-Son, Alexandre Allauzen, and François Yvon (2012b). “Measuring the Influence of Long Range Dependencies with Neural Network Language Models”. In: *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*.

Montréal, Canada: Association for Computational Linguistics, pp. 1–10. URL: http://www.aclweb.org/anthology/W12-2701 (cit. on p. 102).

Le, Hai-Son, Alex Allauzen, Guillaume Wisniewski, and François Yvon (2010). "Training continuous space language models: some practical issues". In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. EMNLP '10. Cambridge, Massachusetts: Association for Computational Linguistics, pp. 778–788. URL: http://dl.acm.org/citation.cfm?id=1870658.1870734 (cit. on pp. 74, 84).

Le, Hai-Son, Ilya Oparin, Abdel. Messaoudi, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon (2011a). "Large Vocabulary SOUL Neural Network Language Models". In: *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011)*. Florence, Italy. URL: http://www.isca-speech.org/archive/interspeech_2011/i11_1469.html (cit. on pp. 61, 62).

Le, Hai-Son, Ilya Oparin, Alex Allauzen, Jean-Luc Gauvain, and François Yvon (2011b). "Structured Output Layer neural network language model". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pp. 5524 –5527. DOI: 10.1109/ICASSP.2011.5947610 (cit. on pp. 55, 62).

Le, Hai-Son, Thomas Lavergne, Alexandre Allauzen, Marianna Apidianaki, Li Gong, Aurélien Max, Artem Sokolov, Guillaume Wisniewski, and François Yvon (2012a). "LIMSI @ WMT12". In: *Proceedings of the Seventh Workshop on Statistical Machine Translation*. Montréal, Canada: Association for Computational Linguistics, pp. 330–337. URL: http://www.aclweb.org/anthology/W12-3141 (cit. on p. 127).

Le, Hai-Son, Iya Oparin, Alex Allauzen, Jean-Luc Gauvain, and François Yvon (2013). "Structured Output Layer Neural Network Language Models for Speech Recognition". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 21.1, pp. 195 –204. ISSN: 1558-7916. DOI: 10.1109/TASL.2012.2215599 (cit. on p. 73).

Le, Quoc V., Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg Corrado, Jeff Dean, and Andrew Ng (2012b). "Building high-level features using large scale unsupervised learning". In: *International Conference in Machine Learning* (cit. on p. 131).

Lecorvé, Gwénolé and Petr Motlicek (09/2012). "Conversion of Recurrent Neural Network Language Models to Weighted Finite State Transducers for Automatic Speech Recognition". In: *Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH 2012)*. Portland, Oregon, USA (cit. on p. 131).

Lopez, Adam (08/2008). "Statistical machine translation". In: *ACM Comput. Surv.* 40.3, 8:1–8:49. ISSN: 0360-0300. DOI: 10.1145/1380584.1380586. URL: http://doi.acm.org/10.1145/1380584.1380586 (cit. on p. 8).

Luo, J., L. Lamel, and J.-L. Gauvain (2009). "Modeling characters versuswords for mandarin speech recognition". In: *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pp. 4325 –4328. DOI: 10.1109/ICASSP.2009.4960586 (cit. on p. 63).

Mariño, José B., Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, José A. R. Fonollosa, and Marta R. Costa-jussà (12/2006). "N-gram-based Machine Translation". In: *Comput. Linguist.* 32.4, pp. 527–549. ISSN: 0891-2017. DOI: 10.1162/coli.2006.32.4.527. URL: http://dx.doi.org/10.1162/coli.2006.32.4.527 (cit. on pp. 119, 120).

Martin, Sven, Jörg Liermann, and Hermann Ney (04/1998). "Algorithms for bigram and trigram word clustering". In: *Speech Commun.* 24.1, pp. 19–37. ISSN: 0167-6393. DOI: 10.1016/S0167-6393(97)00062-9. URL: http://dx.doi.org/10.1016/S0167-6393(97)00062-9 (cit. on pp. 16, 24).

Miikkulainen, Risto and Michael G. Dyer (1991). "Natural Language Processing with Modular Neural Networks and Distributed Lexicon". In: *Cognitive Science* 15, pp. 343–399 (cit. on p. 30).

Mikolov, Tomas, Anoop Deoras, Stefan Kombrink, Lukás Burget, and Jan Cernocký (2011a). "Empirical Evaluation and Combination of Advanced Language Modeling Techniques". In: *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011)*, pp. 605–

608. URL: http://www.isca-speech.org/archive/interspeech_2011/i11_0605.html (cit. on pp. 3, 37).

Mikolov, Tomáš, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur (2010). "Recurrent neural network based language model". In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*. Vol. 2010. 9. Makuhari, Chiba, Japan, pp. 1045–1048. URL: http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html (cit. on pp. 30, 37, 39, 112).

Mikolov, Tomáš, Stefan Kombrink, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur (2011b). "Extensions of recurrent neural network language model". In: *Proc. of ICASSP'11*, pp. 5528–5531 (cit. on pp. 39, 74, 109, 113).

Mikolov, Tomáš, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černocký (2011c). "Strategies for Training Large Scale Neural Network Language Models". In: *Proceedings of ASRU 2011*. Hawaii, US: IEEE Signal Processing Society, pp. 196–201. ISBN: 978-1-4673-0366-8. URL: http://www.fit.vutbr.cz/research/view_pub.php?id=9775 (cit. on pp. 39, 57, 60, 75, 101, 111, 113).

Mnih, Andriy and G. E. Hinton (2008). "A Scalable Hierarchical Distributed Language Model". In: *Advances in Neural Information Processing Systems 21*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Vol. 21, pp. 1081–1088 (cit. on pp. 30, 35, 37, 55, 57).

Mnih, Andriy and Geoffrey Hinton (2007). "Three new graphical models for statistical language modelling". In: *Proceedings of the 24th international conference on Machine learning*. ICML '07. Corvalis, Oregon: ACM, pp. 641–648. ISBN: 978-1-59593-793-3. DOI: 10.1145/1273496.1273577. URL: http://doi.acm.org/10.1145/1273496.1273577 (cit. on pp. 30, 34, 83, 99).

Mohamed, A., G.E. Dahl, and G. Hinton (2012). "Acoustic Modeling Using Deep Belief Networks". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 20.1, pp. 14 –22. ISSN: 1558-7916. DOI: 10.1109/TASL.2011.2109382 (cit. on pp. 74, 130).

Morin, F. and Y. Bengio (2005). "Hierarchical probabilistic neural network language model". In: *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pp. 246–252 (cit. on pp. 35, 55, 57).

Mrva, David and Philip C. Woodland (2004). "A PLSA-based language model for conversational telephone speech". In: *Proceedings of the 8th International Conference on Spoken Language Processing (INTERSPEECH 2004 - ICSLP)*. Jeju Island, Korea, pp. 2257–2260. URL: http://www.isca-speech.org/archive/interspeech_2004/i04_2257.html (cit. on pp. 15, 21).

Nakamura, Masami, Katsuteru Maruyama, Takeshi Kawabata, and Kiyohiro Shikano (1990). "Neural network approach to word category prediction for English texts". In: *Proceedings of the 13th conference on Computational linguistics - Volume 3*. COLING '90. Helsinki, Finland: Association for Computational Linguistics, pp. 213–218. ISBN: 952-90-2028-7. DOI: 10.3115/991146.991184. URL: http://dx.doi.org/10.3115/991146.991184 (cit. on p. 30).

Ney, Hermann, Ute Essen, and Reinhard Kneser (1994). "On structuring probabilistic dependences in stochastic language modelling". In: *Computer Speech & Language* 8.1, pp. 1 –38. ISSN: 0885-2308. DOI: 10.1006/csla.1994.1001. URL: http://www.sciencedirect.com/science/article/pii/S0885230884710011 (cit. on p. 12).

Niesler, Thomas R. (1997). "Category-based statistical language models". PhD thesis. University of Cambridge (cit. on pp. xviii, 15).

Och, Franz Josef (2003). "Minimum error rate training in statistical machine translation". In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. ACL '03. Sapporo, Japan: Association for Computational Linguistics, pp. 160–167. DOI: 10.3115/1075096.1075117. URL: http://dx.doi.org/10.3115/1075096.1075117 (cit. on pp. 6, 49, 70, 113, 125).

Och, Franz Josef and Hermann Ney (03/2003). "A systematic comparison of various statistical alignment models". In: *Comput. Linguist.* 29.1, pp. 19–51. ISSN: 0891-2017. DOI: 10.1162/089120103321337421. URL: http://dx.doi.org/10.1162/089120103321337421 (cit. on p. 124).

Och, Franz Josef and Hermann Ney (12/2004). "The Alignment Template Approach to Statistical Machine Translation". In: *Comput. Linguist.* 30 (4), pp. 417–449. ISSN: 0891-2017. DOI: http://dx.doi.org/10.1162/0891201042544884. URL: http://dx.doi.org/10.1162/0891201042544884 (cit. on p. 118).

Oparin, I., L. Lamel, and J.-L. Gauvain (2010a). "Improving Mandarin Chinese STT system with Random Forests language models". In: *Chinese Spoken Language Processing (ISCSLP), 2010 7th International Symposium on*, pp. 242 –245. DOI: 10.1109/ISCSLP.2010.5684903 (cit. on pp. 62, 63).

Oparin, I., O. Glembek, L. Burget, and J. Cernocky (2008). "Morphological random forests for language modeling of inflectional languages". In: *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE*, pp. 189 –192. DOI: 10.1109/SLT.2008.4777872 (cit. on pp. xviii, 15, 25).

Oparin, Ilya, Lori Lamel, and Jean-Luc Gauvain (2010b). "Large-Scale Language Modeling with Random Forests for Mandarin Chinese Speech-to-Text". In: *Advances in Natural Language Processing*. Ed. by Hrafn Loftsson, Eiríkur Rögnvaldsson, and Sigrún Helgadóttir. Vol. 6233. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 269–280. ISBN: 978-3-642-14769-2. URL: http://dx.doi.org/10.1007/978-3-642-14770-8_31 (cit. on pp. 24, 25).

Oparin, Ilya, Martin Sundermeyer, Herman Ney, and Jean-Luc Gauvain (2012). "Performance analysis of Neural Networks in combination with n-gram language models". In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 5005 –5008. DOI: 10.1109/ICASSP.2012.6289044 (cit. on p. 131).

Paccanaro, Alberto and Geoffrey E. Hinton (03/2001). "Learning Distributed Representations of Concepts Using Linear Relational Embedding". In: *IEEE Trans. on Knowl. and Data Eng.* 13.2, pp. 232–244. ISSN: 1041-4347. DOI: 10.1109/69.917563. URL: http://dx.doi.org/10.1109/69.917563 (cit. on p. 30).

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: http://dx.doi.org/10.3115/1073083.1073135 (cit. on p. 8).

Park, Junho, Xunying Liu, Mark J. F. Gales, and Philip C. Woodland (2010). "Improved neural network based language modelling and adaptation". In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*. Makuhari, Chiba, Japan: International Speech Communication Association, pp. 1041–1044. URL: http://www.isca-speech.org/archive/interspeech_2010/i10_1041.html (cit. on p. 48).

Potamianos, Gerasimos and Frederick Jelinek (1998). "A study of n-gram and decision tree letter language modeling methods". In: *Speech Communication* 24.3, pp. 171 –192. ISSN: 0167-6393. DOI: 10.1016/S0167-6393(98)00018-1. URL: http://www.sciencedirect.com/science/article/pii/S0167639398000181 (cit. on pp. 23, 58).

Radinsky, Kira, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch (2011). "A word at a time: computing word relatedness using temporal semantic analysis". In: *Proceedings of the 20th international conference on World wide web*. WWW '11. Hyderabad, India: ACM, pp. 337–346. ISBN: 978-1-4503-0632-4. DOI: 10.1145/1963405.1963455. URL: http://doi.acm.org/10.1145/1963405.1963455 (cit. on pp. 95, 97).

Rastrow, Ariya, Sanjeev Khudanpur, and Mark Dredze (2012). "Revisiting the Case for Explicit Syntactic Information in Language Models". In: *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. Montréal, Canada: Association for Computational Linguistics, pp. 50–58. URL: http://www.aclweb.org/anthology/W12-2707 (cit. on p. 18).

Roark, Brian Edward (2001). "Robust probabilistic predictive syntactic processing: motivations, models, and applications". AAI3006783. PhD thesis. Providence, RI, USA. ISBN: 0-493-16094-9 (cit. on p. 17).

Rosenfeld, R. (2000). “Two decades of statistical language modeling: where do we go from here?” In: *Proceedings of the IEEE* 88.8, pp. 1270 –1278. ISSN: 0018-9219. DOI: 10.1109/5.880083 (cit. on pp. 3, 7, 101).

Rosenfeld, Ronald (1994). “Adaptive Statistical Language Modeling: A Maximum Entropy Approach”. PhD thesis. Carnegie Mellon University (cit. on pp. 25, 105).

Rosenfeld, Ronald (1996). “A maximum entropy approach to adaptive statistical language modelling”. In: *Computer Speech & Language* 10.3, pp. 187 –228. ISSN: 0885-2308. DOI: 10.1006/csla.1996.0011. URL: http://www.sciencedirect.com/science/article/pii/S088523089690011X (cit. on p. 15).

Rosenfeld, Ronald, Stanley F. Chen, and Xiaojin Zhu (2001). “Whole-sentence exponential language models: a vehicle for linguistic-statistical integration”. In: *Computer Speech & Language* 15.1, pp. 55 –73. ISSN: 0885-2308. DOI: 10.1006/csla.2000.0159. URL: http://www.sciencedirect.com/science/article/pii/S0885230800901591 (cit. on p. 101).

Roth, Ryan, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin (2008). “Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking”. In: *Proceedings of ACL-08: HLT, Short Papers*. Columbus, Ohio: Association for Computational Linguistics, pp. 117–120. URL: http://www.aclweb.org/anthology/P/P08/P08-2030 (cit. on pp. 64, 93).

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). “Parallel distributed processing: explorations in the microstructure of cognition, vol. 1”. In: ed. by David E. Rumelhart and James L. McClelland. Cambridge, MA, USA: MIT Press. Chap. Learning internal representations by error propagation, pp. 318–362. URL: http://dl.acm.org/citation.cfm?id=104279.104293 (cit. on pp. 39, 109).

Sarikaya, Ruhi, Yonggang Deng, Mohamed Afify, Brian Kingsbury, and Yuqing Gao (2008). “Machine translation in continuous space”. In: *Proceedings of the 9th Annual Conference of the International Speech Communication Association (INTERSPEECH 2008)*. Brisbane, Australia, pp. 2350–2353. URL: http://www.isca-speech.org/archive/interspeech_2008/i08_2350.html (cit. on p. 118).

Schmid, Helmut (1994). “Probabilistic Part-of-Speech Tagging Using Decision Trees”. In: *Proceedings of International Conference on New Methods in Language Processing* (cit. on pp. 106, 124).

Schmidhuber, J. and S. Heil (1996). “Sequential neural text compression”. In: *Neural Networks, IEEE Transactions on* 7.1, pp. 142 –146. ISSN: 1045-9227. DOI: 10.1109/72.478398 (cit. on p. 30).

Schwartz, Lane, Chris Callison-Burch, William Schuler, and Stephen Wu (2011). “Incremental Syntactic Language Models for Phrase-based Translation”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 620–631. URL: http://www.aclweb.org/anthology/P11-1063 (cit. on p. 101).

Schwenk, H. and J.-L. Gauvain (2004). “Neural Network Language Models for Conversational Speech Recognition”. In: *International Conference on Speech and Language Processing*, pp. 1215–1218 (cit. on p. 48).

Schwenk, H. and P. Koehn (2008). “Large and Diverse Language Models for Statistical Machine Translation”. In: *International Joint Conference on Natural Language Processing*, pp. 661–666 (cit. on p. 101).

Schwenk, Holger (07/2007). “Continuous space language models”. In: *Comput. Speech Lang.* 21.3, pp. 492–518. ISSN: 0885-2308. DOI: http://dx.doi.org/10.1016/j.csl.2006.09.003 (cit. on pp. xviii, 30, 31, 47, 48, 63, 103, 129).

Schwenk, Holger, Daniel Dchelotte, and Jean-Luc Gauvain (2006). “Continuous space language models for statistical machine translation”. In: *Proceedings of the COLING/ACL on Main conference poster sessions*. COLING-ACL ’06. Sydney, Australia: Association for Computational Linguistics, pp. 723–730. URL: http://dl.acm.org/citation.cfm?id=1273073.1273166 (cit. on pp. xix, 30).

Schwenk, Holger and Jean-Luc Gauvain (2002). “Connectionist language modeling for large vocabulary continuous speech recognition”. In: *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*. Vol. 1, pp. I–765 –I–768. DOI: 10.1109/ICASSP.2002.5743830 (cit. on pp. 47, 55).

Schwenk, Holger, Marta R. Costa-jussa, and Jose A. R. Fonollosa (2007). “Smooth Bilingual *N*-Gram Translation”. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 430–438. URL: http://www.aclweb.org/anthology/D/D07/D07-1045 (cit. on pp. 117, 119).

Schwenk, Holger, Anthony Rousseau, and Mohammed Attik (2012). “Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation”. In: *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. Montréal, Canada: Association for Computational Linguistics, pp. 11–19. URL: http://www.aclweb.org/anthology/W12-2702 (cit. on pp. 48, 131).

Seide, Frank, Gang Li, and Dong Yu (2011). “Conversational Speech Transcription Using Context-Dependent Deep Neural Networks”. In: *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011)*, pp. 437–440. URL: http://www.isca-speech.org/archive/interspeech_2011/i11_0437.html (cit. on pp. 74, 130).

Shannon, C. E. (1948). “A mathematical theory of communication”. In: *Bell system technical journal* 27 (cit. on p. 5).

Shannon, C. E. (1951). “Prediction and entropy of printed english.” In: *Bell Systems Technical Journal*, pp. 50–64 (cit. on p. 7).

Shaoul, Cyrus and Chris Westbury (2010). “The Westbury Lab Wikipedia Corpus”. In: Edmonton, AB: University of Alberta. URL: http://www.psych.ualberta.ca/~westburylab/downloads/westburylab.wikicorp.download.html (cit. on p. 96).

Socher, Richard, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning (2011). “Parsing Natural Scenes and Natural Language with Recursive Neural Networks”. In: *Proceedings of the 26th International Conference on Machine Learning (ICML)* (cit. on p. 131).

Teh, Yee Whye (2006). “A Hierarchical Bayesian Language Model Based On Pitman-Yor Processes”. In: *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia: Association for Computational Linguistics, pp. 985–992. DOI: 10.3115/1220175.1220299. URL: http://www.aclweb.org/anthology/P06-1124 (cit. on p. 11).

Tillmann, Christoph (2004). “A unigram orientation model for statistical machine translation”. In: *Proceedings of HLT-NAACL 2004: Short Papers*. HLT-NAACL-Short ’04. Boston, Massachusetts: Association for Computational Linguistics, pp. 101–104. ISBN: 1-932432-24-8. URL: http://dl.acm.org/citation.cfm?id=1613984.1614010 (cit. on p. 124).

Turian, Joseph, Lev Ratinov, and Yoshua Bengio (2010). “Word representations: a simple and general method for semi-supervised learning”. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. ACL ’10. Uppsala, Sweden: Association for Computational Linguistics, pp. 384–394. URL: http://dl.acm.org/citation.cfm?id=1858681.1858721 (cit. on p. 94).

Turing, A. M. (1950). *Computing Machinery and Intelligence*. One of the most influential papers in the history of the cognitive sciences: http://cogsci.umn.edu/millennium/final.html (cit. on p. xviii).

Utgoff, P.E. and D.J. Stracuzzi (2002). “Many-layered learning”. In: *Development and Learning, 2002. Proceedings. The 2nd International Conference on*, pp. 141 –146. DOI: 10.1109/DEVLRN.2002.1011824 (cit. on p. 73).

Watanabe, Yotaro, Masayuki Asahara, and Yuji Matsumoto (2007). “A Graph-Based Approach to Named Entity Categorization in Wikipedia Using Conditional Random Fields”. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language*

Learning (EMNLP-CoNLL). Prague, Czech Republic: Association for Computational Linguistics, pp. 649–657. URL: http://www.aclweb.org/anthology/D/D07/D07-1068 (cit. on p. 6).

Xu, P., S. Khudanpur, M. Lehr, E. Prud'hommeaux, N. Glenn, D. Karakos, B. Roark, K. Sagae, M. Saraclar, I. Shafran, D. Bikel, C. Callison-Burch, Y. Cao, K. Hall, E. Hasler, P. Koehn, A. Lopez, M. Post, and D. Riley (2012). "Continuous space discriminative language modeling". In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 2129 –2132. DOI: 10.1109/ICASSP.2012.6288332 (cit. on p. 131).

Xu, Peng and Frederick Jelinek (2004). "Random Forests in Language Modeling". In: *Proceedings of EMNLP 2004*. Ed. by Dekang Lin and Dekai Wu. Barcelona, Spain: Association for Computational Linguistics, pp. 325–332 (cit. on pp. 15, 22).

Xu, Peng and Frederick Jelinek (2007). "Random forests and the data sparseness problem in language modeling". In: *Computer Speech & Language* 21.1, pp. 105 –152. ISSN: 0885-2308. DOI: 10.1016/j.csl.2006.01.003. URL: http://www.sciencedirect.com/science/article/pii/S0885230806000040 (cit. on p. 58).

Xu, Wei and Alex Rudnicky (2000). "Can artificial neural networks learn language models?" In: *Proceedings of the 6th International Conference on Spoken Language Processing, ICSLP 2000*, pp. 202–205 (cit. on p. 30).

Yih, Wen-tau and Vahed Qazvinian (2012). "Measuring Word Relatedness Using Heterogeneous Vector Space Models". In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montréal, Canada: Association for Computational Linguistics, pp. 616–620. URL: http://www.aclweb.org/anthology/N12-1077 (cit. on p. 95).

Zamora-Martinez, Francisco, Maria José Castro-Bleda, and Holger Schwenk (2010). "N-gram-based Machine Translation enhanced with Neural Networks for the French-English BTEC-IWSLT'10 task". In: *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*. Paris, France, pp. 45–52 (cit. on p. 117).

Zens, Richard, Franz Josef Och, and Hermann Ney (2002). "Phrase-Based Statistical Machine Translation". In: *KI '02: Proceedings of the 25th Annual German Conference on AI*. London, UK: Springer-Verlag, pp. 18–32. ISBN: 3-540-44185-9 (cit. on pp. 118, 121).

Zipf, G. K. (1932). *Selective Studies and the Principle of Relative Frequency in Language* (cit. on pp. 10, 29, 112).

Zweig, Geoffrey and Chris J.C. Burges (2012). "A Challenge Set for Advancing Language Modeling". In: *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. Montréal, Canada: Association for Computational Linguistics, pp. 29–36. URL: http://www.aclweb.org/anthology/W12-2704 (cit. on p. 7).