Apprentissage discriminant des modèles continus en traduction automatique

Quoc Khanh Quoc Khanh Do

NNT : 2016SACL5071

THÈSE DE DOCTORAT
DE
L'UNIVERSITÉ PARIS-SACLAY
PRÉPARÉE À
L'UNIVERSITÉ PARIS-SUD

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Spécialité de doctorat : Informatique

Par

M. Quoc Khanh Do

Apprentissage Discriminant des Modèles Continus en Traduction Automatique

Thèse présentée et soutenue à Orsay, le 31/03/2016

Composition du Jury :

Rapporteur	:	M. Thierry Artières	Professeur, Université de Marseille
Rapporteur	:	M. Christof Monz	Associate Professor, University of Amsterdam
Examinateur	:	M. Holger Schwenk (président du jury)	Facebook Artificial Intelligence Research Paris
Examinatrice	:	Mme. Laurence Likforman-Sulem	Maître de Conférence, Télécom ParisTech
Directeur de thèse	:	M. François Yvon	Professeur, Université Paris-Sud
Co-encadrant	:	M. Alexandre Allauzen	Maître de Conférence, Université Paris-Sud

Gửi tặng ba Đỗ Thanh, mẹ Nguyễn Thị Ba, em gái Đỗ Thị Ly
và người bạn đời tôi hết mực yêu thương, Huỳnh Thị Thanh Huyền

Acknowledgements

I would like to express my gratitude to François Yvon and Alexandre Allauzen, my thesis supervisors, for their constant guidance and encouragement during my three years at LIMSI-CNRS. My research as well as my critical thinking skills have been improving significantly thanks to the meetings and seminars I participated with them. Their devotion has helped me to overcome the most difficult moments in the thesis preparation.

I am also grateful to the other members of the jury : Thierry Artière, Christof Monz, Laurence Likforman-Sulem, and Holger Schwenk for their questions, analyses and suggestions. My special thanks to Thierry Artière and Christof Monz for spending a lot of time and efforts to review this thesis, and to Holger Schwenk for his report of the defense.

Ideas and directions for my research have also been raised from discussions with other TLP group members. LIMSI laboratory has offered an ideal and joyful working environment where the ideas of each individual are all considered, and which helps us to combine our forces from different research directions. The WMT evaluation campaign each year has been an occasion for me to benefit from the knowledge of my dear friends. My appreciation to Nicolas Pécheux who introduced me to n-code and data pre-processing procedures for Machine Translation. MOSES was explained to me by Benjamin Marie (though he was not a member of TLP) along with several tuning algorithms that were crucially important at some points in my thesis. Matthieu Labeau presented the first implementation of NCE at LIMSI which persuaded me that this method is worthwhile. Discriminative large-margin methods were first explained to me by Alexandre Allauzen who had nurtured the ideas for a long time. Other members have made my life as a PhD student much more easier even though they were not directly implied in my work. Li Gong in the first time, and Elena Knyazeva later are the "chefs" of our bureau who made our days more comfortable with their kindness and their humour. Yong Xu discussed with me about China during lunch times. Thomas Lavergne helped me to understand other aspects of the research career. Along with Hélène Maynard, I have had unforgettable moments discovering Lake Tahoe area during the very first international scientific conference of my career. I also had fruitful discussions with funny stories and jokes with other limsians : Marianna Apidianki, Claude Barras, Philippe Boula De Mareüil, Hervé Bredin, Guillaume Wisniewski, Lauriane Aufrant, Rachel Bawden (she impressed me with her dynamic characteristics although we only met in the last several months), Julia Ive, Kevin Loser, Franck Burlot, Rasa Lileikyté... Many thanks for all the great moments we shared together.

I would like to thank Hai Son Le for his SOUL source codes from which I learnt a lot about C++ programming language, and on which I implemented my research ideas. He was also the person who convinced me to do my PhD thesis at LIMSI on the field which I am still working on. Also, I thank Jean-Luc Gauvain for allowing me to intensively use

cluster machines.

My deepest appreciation to the used-to-be-girlfriend and now my wife, whose richly emotional life makes my life happier even in the toughest moments. And my limitless gratitude to my parents and my sister who always stay beside me and support me.

Abstract

Over the past few years, neural network (NN) architectures have been successfully applied to many Natural Language Processing (NLP) applications, such as Automatic Speech Recognition (ASR) and Statistical Machine Translation (SMT). For the language modeling task, these models consider linguistic units (*i.e* words and phrases) through their projections into a continuous (multi-dimensional) space, and the estimated distribution is a function of these projections. Also qualified *continuous-space models* (CSMs), their peculiarity hence lies in this exploitation of a continuous representation that can be seen as an attempt to address the sparsity issue of the conventional discrete models.

In the context of SMT, these techniques have been applied on neural network-based language models (NNLMs) included in SMT systems, and on continuous-space translation models (CSTMs). These models have led to significant and consistent gains in the SMT performance, but are also considered as very expensive in training and inference, especially for systems involving large vocabularies. To overcome this issue, Structured Output Layer (SOUL) and Noise Contrastive Estimation (NCE) have been proposed; the former modifies the standard structure on vocabulary words, while the latter approximates the maximum-likelihood estimation (MLE) by a sampling method. All these approaches share the same estimation criterion which is the MLE; however using this procedure results in an inconsistency between the objective function defined for parameter estimation and the way models are used in the SMT application.

The work presented in this dissertation aims to design new performance-oriented and global training procedures for CSMs to overcome these issues. The main contributions lie in the investigation and evaluation of efficient training methods for (large-vocabulary) CSMs which aim :(a) to reduce the total training cost, and (b) to improve the efficiency of these models when used within the SMT application. On the one hand, the training and inference cost can be reduced (using the SOUL structure or the NCE algorithm), or by reducing the number of iterations via a faster convergence. This thesis provides an empirical analysis of these solutions on different large-scale SMT tasks. On the other hand, we propose a discriminative training framework which optimizes the performance of the whole system containing the CSM as a component model. The experimental results show that this framework is efficient to both train and adapt CSM within SMT systems, opening promising research perspectives.

Key words: Statistical Machine Translation, Neural Network, Continuous-Space Models, Discriminative Training, Large-Margin Methods, Noise Contrastive Estimation

Résumé

Durant ces dernières années, les architectures de réseaux de neurones (RN) ont été appliquées avec succès à de nombreuses applications en Traitement Automatique de Langues (TAL), comme par exemple en Reconnaissance Automatique de la Parole (RAP) ainsi qu'en Traduction Automatique (TA). Pour la tâche de modélisation statique de la langue, ces modèles considèrent les unités linguistiques (c'est-à-dire des mots et des segments) à travers leurs projections dans un espace continu (multi-dimensionnel), et la distribution de probabilité à estimer est une fonction de ces projections. Ainsi connus sous le nom de *modèles continus* (MC), la particularité de ces derniers se trouve dans l'exploitation de la représentation continue qui peut être considérée comme une solution au problème de données creuses rencontré lors de l'utilisation des modèles discrets conventionnels.

Dans le cadre de la TA, ces techniques ont été appliquées dans les modèles de langue neuronaux (MLN) utilisés dans les systèmes de TA, et dans les modèles continus de traduction (MCT). L'utilisation de ces modèles se sont traduit par d'importantes et significatives améliorations des performances des systèmes de TA. Ils sont néanmoins très coûteux lors des phrases d'apprentissage et d'inférence, notamment pour les systèmes ayant un grand vocabulaire.

Afin de surmonter ce problème, l'architecture SOUL (pour *Structured Output Layer* en anglais) et l'algorithme NCE (pour *Noise Contrastive Estimation*, ou l'estimation contrastive bruitée) ont été proposés: le premier modifie la structure standard de la couche de sortie, alors que le second cherche à approximer l'estimation du maximum de vraisemblance (MV) par une méthode d'échantillonnage. Toutes ces approches partagent le même critère d'estimation qui est la log-vraisemblance; pourtant son utilisation mène à une incohérence entre la fonction objectif définie pour l'estimation des modèles, et la manière dont ces modèles seront utilisés dans les systèmes de TA.

Cette dissertation vise à concevoir de nouvelles procédures d'entraînement des MC, afin de surmonter ces problèmes. Les contributions principales se trouvent dans l'investigation et l'évaluation des méthodes d'entraînement efficaces pour MC qui visent à: (i) réduire le temps total de l'entraînement, et (ii) améliorer l'efficacité de ces modèles lors de leur utilisation dans les systèmes de TA. D'un côté, le coût d'entraînement et d'inférence peut être réduit (en utilisant l'architecture SOUL ou l'algorithme NCE), ou la convergence peut être accélérée. La dissertation présente une analyse empirique de ces approches pour des tâches de traduction automatique à grande échelle. D'un autre côté, nous proposons un cadre d'apprentissage discriminant qui optimise la performance du système entier ayant incorporé un modèle continu. Les résultats expérimentaux montrent que ce cadre d'entraînement est efficace pour l'apprentissage ainsi que pour l'adaptation des MC au sein des systèmes de TA, ce qui ouvre de nouvelles perspectives prometteuses.

Mots clefs: Traduction Automatique Statistique, Réseau de neurones, Modèles Continus de Traduction, Apprentissage Discriminant, Méthodes à Larges Marges, Estimation Contrastive Bruitée

Contents

List of Figures

List of Tables

Introduction

Over the past few years, research on neural network (NN) architectures for Natural Language Processing has been rejuvenated with the pioneering work of (Bengio et al., 2001, 2003a). Boosted by early successes in the language modelling (LM) task for Automatic Speech Recognition (ASR) (Schwenk and Gauvain, 2002; Schwenk, 2007; Mnih and Hinton, 2008; Le et al., 2011), NNs have since been successfully applied to many other tasks, such as syntactic analysis (Socher et al., 2013), estimation of semantic similarity (Huang et al., 2012a), word alignment (Yang et al., 2013), acoustic modelling in ASR (Seide et al., 2011; Dahl et al., 2012), or in Statistical Machine Translation (SMT) (Le et al., 2012a; Kalchbrenner and Blunsom, 2013; Devlin et al., 2014; Cho et al., 2014).

The peculiarity of these models lies in their exploitation of continuous representations of linguistic units, such as words or phrases. Indeed, traditional probabilistic models are based on a discrete representation of these units, which considers every unit as the realizations of a discrete random variable. As a result, estimation is essentially a *counting operation*, which does not embody any notion of similarity between the different events. As a typical example, when estimating the likelihood of word sequences in an English corpus, the procedure collects the occurrences of different morphological forms of some verbs (for instance, *eat*, *eats* and *ate*) as being unrelated outcomes; the occurrences of each form do not contribute any information to the estimation of the others. The peculiar distribution of words in texts leads to the fact that the counting is always carried out on a too small number of occurrences, hence is not statistically reliable and weakly generalizable. On the other hand, neural network-based models, also qualified as *Continuous-space models* (CSMs), consider these units through their projections into a continuous (multi-dimensional) space. The estimated distribution is a function of these projections (also called *embeddings*). These vectors, along with the function parameters, are jointly trained in the neural network framework; the unified training procedure has the effect that the units appearing in similar contexts have similar embeddings. As a result, a notion of similarity between units can be induced, which leads to a better exploitation of linguistic corpora.

In the context of SMT, these techniques have been applied, first on neural network-based language models (NNLMs) included in SMT systems, then on bilingual continuous-space translation models (CSTMs) (Schwenk et al., 2007; Le et al., 2012a). These models have led to significant and consistent gains in the SMT performance (Le et al., 2012a; Devlin et al., 2014; Do et al., 2014c), but are also considered as very expensive in training and inference, especially for systems involving large vocabularies. The bottleneck has been identified since the beginning : conventional output layers require to compute the score for every vocabulary words (or linguistic units) for a normalization via a *softmax*

layer (Bengio et al., 2001; Schwenk and Gauvain, 2004). To overcome this issue, the Structured Output Layer (SOUL) has been proposed (Le et al., 2011, 2013); its building however requires to modify the standard architecture with a structured output layer on vocabulary words. Instead, approximate procedures can be used to train the standard structure (Bengio et al., 2003b; Bengio and Senécal, 2008), but the resulting estimation is often very unstable. *Noise Contrastive Estimation* (NCE), described in (Gutmann and Hyvärinen, 2010) and applied to NNLMs in (Mnih and Teh, 2012), might be a more computationally efficient alternative, but its interpretation is difficult and its dependence on a *noise* distribution still needs investigation.

All these approaches share (directly or not) the same estimation criterion which is the maximum-likelihood estimation (MLE). However, using this procedure results in an inconsistency between the objective function defined for parameter estimation and the way models are used in the SMT application. The same issue has been identified in the training of traditional language models for which the usual evaluation by *perplexity* is known to only loosely correlate with the performance of the application for which LMs have been designed. For Speech Recognition systems, (Rosenfeld, 2000) claims that a reduction of 5% in perplexity is usually not practically significant; a $10-20\%$ reduction is noteworthy, while only a perplexity improvement of 30% or more is significant and can be translated into performance improvement of the ASR. The MLE training for CSMs has the same limitation. In particular, it is solely based on the CSM itself, and does not take into account the properties of the SMT system into which the model will be later incorporated.

Recent developments of large-scale discriminative SMT (Cherry and Foster, 2012; Yu et al., 2013; Zhao et al., 2014) have put forward a set of training criteria which have been designed to closely correlate with MT quality metrics (such as BLEU), and of training procedures that can optimize the whole system in a unified manner. Ideas from this line of research are henceforth promising for the design of new performance-oriented and global training procedures for CSMs, which constitute one of the main motivations for the work described in this dissertation.

In this context, the main contributions of this dissertation lie in the investigation and evaluation of efficient training methods for (large-vocabulary) continuous-space models (CSMs) which aim : (a) to reduce the total training cost, and (b) to improve the efficiency of these models when used within the SMT application. On the one hand, (a) can be obtained either by reducing the cost of each training iteration (using the SOUL structure or the NCE algorithm), or by reducing the number of iterations for a faster convergence (via a better learning rate adaptation). On the other hand, (b) is achieved by proposing a discriminative training framework which optimizes the performance of the whole system containing the CSM as a component model.

The dissertation will be organized as follows. Chapter 1 describes various SMT systems in which a CSM can be used, and introduces training criteria that are widely used in the discriminative training and tuning of translation systems. Chapter 2 reviews several state-of-the-art neural network structures used in NNLMs and CSTMs. The next two chapters discuss the MLE training, comparing the exact optimization with SOUL with an approximate training procedure based on NCE (Chapter 3), and proposing several adaptive learning rate methods to speed up convergence and reduce the number of

training iterations (Chapter 4). A discriminative training framework, which corrects the weaknesses related to the MLE will be introduced in Chapter 5. We will also investigate various aspects of this framework, along with an intelligent combination with the NCE approach.

1 Statistical Machine Translation

This chapter provides a description of several approaches used in Statistical Machine Translation (SMT), and of the *log-linear model* into which continuous-space models (CSMs) will be incorporated. These discussions give the foundation, not only for the integration, but also for the training of CSMs that will be described in the next chapters. In the first section, we give a brief overview on several widely used SMT systems, in particular the phrase-based and n-gram-based approaches. Some methods used to evaluate the quality of SMT outputs are presented in Section 1.2. The final section formulates and describes the discriminative training of log-linear coefficients, along with the corresponding optimization algorithms. Following (Cherry and Foster, 2012), several discriminative objective functions used in these methods are reviewed (Section 1.3.1), which lay the ground for further developments.

1.1 Overview on Statistical Machine Translation

Translation from a source sentence $\mathbf{s}$ to a target sentence $\mathbf{t}$ is a complex process usually modelled by dividing the whole derivation into smaller sub-processes, such as the segmentation of the source text $\mathbf{s}$ into smaller fragments, the translation of each individual fragment into target words, and their final recombination. Each small sub-process can be modelled independently and is learnt using the statistical analysis of large corpora, while the interaction between different models is also learnt.

Early work on SMT considers the translation as a generative process. The modelling aims at estimating the probability of the target sentence given its corresponding source, and involves probabilistic distributions at the word level. The first mathematically solid approach to the modelling of the translation process is fully described in (Brown et al.,

1993), which considers translation as a search for the most likely target sentence $\mathbf{t}^*$:

$$\mathbf{t}^* = \underset{\mathbf{t}}{\operatorname{argmax}}\, \mathbf{p}(\mathbf{t}|\mathbf{s})$$

The solution of (Brown et al., 1993), known in the literature as the noisy channel model (or source channel model), is to apply the Bayes rules to obtain the following decomposition :

$$\begin{aligned}\mathbf{t}^* &= \underset{\mathbf{t}}{\operatorname{argmax}}\, \mathbf{p}(\mathbf{t}|\mathbf{s}) \\ &= \underset{\mathbf{t}}{\operatorname{argmax}}\, \mathbf{p}(\mathbf{s},\mathbf{t}) \\ &= \underset{\mathbf{t}}{\operatorname{argmax}}\, \mathbf{p}(\mathbf{s}|\mathbf{t}) \times \mathbf{p}(\mathbf{t}) \end{aligned} \tag{1.1}$$

This formulation divides the generation of the couple $(\mathbf{s},\mathbf{t})$ in two terms : the choice of a target sentence $\mathbf{t}$ according to a language model (probability $\mathbf{p}(\mathbf{t})$), and the fitting of this sentence compared to source text $\mathbf{s}$ according to a transmission channel modelled by $\mathbf{p}(\mathbf{s}|\mathbf{t})$, also referred to as the translation model. It implies the division of modelling in two subtasks which can be addressed independently in terms of estimation techniques and training data : the language modelling and the translation modelling.

The translation model is estimated on a set of sentence pairs (or parallel training data), each of which contains a sentence in the source language and its translation in the target language. However, these texts present only surface sentence-level relationship without any explicit indication about which decisions have been taken to derive the target sentence from the source. The learning of hidden relationships inside a sentence pair becomes possible only with the introduction of latent variables $\mathbf{a}$ describing the alignment between the components of the source and target sentences. The definition of these latent variables characterizes different approaches in Machine Translation, among which the phrase-based approach (Section 1.1.2) which models the translation based on phrases instead of words, is today one of the state-of-the-art approaches.

The noisy channel model however presents some limitations. First, the decision in the noisy channel model (1.1) can be shown to be optimal only when true probability distributions are used. In practice, distributions are estimated from training data using some simplifying assumptions. These make the models only poor approximations of the assumed data distribution; on the other hand the use of these assumptions is necessary to make statistical estimation tractable. Second, the noisy channel model makes the integration of complex forms of latent variables, as well as of different approaches for the same model, difficult. Later, since the work of (Och and Ney, 2002), the discriminative framework based on a *log-linear* model is often used. This framework models existing knowledge about the translation process (such as the likelihood of a target sentence or of a translation decision) in form of feature functions. In the general case, the probability of $\mathbf{p}(\mathbf{t},\mathbf{a}|\mathbf{s})$ is inferred from the following formula :

$$\mathbf{p}(\mathbf{t},\mathbf{a}|\mathbf{s}) = \frac{1}{\mathrm{H}(\mathbf{s})} \exp\left(\sum_{m=1}^{M} \lambda_m f_m(\mathbf{s},\mathbf{t},\mathbf{a})\right) \tag{1.2}$$

which contains a set of M feature functions reflecting different aspects of the translation decision. Here $\mathrm{H}(\mathbf{s})$ is a normalization constant ensuring all probabilities sum to 1. The

SMT system then selects the target sentence $\mathbf{t}$ (and a derivation $\mathbf{a}$) which maximizes the above probability : [1]

$$\mathbf{t}^* = \underset{\mathbf{t},\mathbf{a}}{\operatorname{argmax}} \sum_{m=1}^{M} \lambda_m f_m(\mathbf{s}, \mathbf{t}, \mathbf{a}) \tag{1.3}$$

It is important to note that the noisy channel model (1.1) is a special case of this discriminative framework, where the set of features contains two functions : the first one reflects the compatibility between $\mathbf{s}$ and $\mathbf{t}$, while the second one evaluates the fluency of $\mathbf{t}$ in the target language; these two are equally weighted with $\lambda_1 = \lambda_2 = 1$. In general, the set of feature functions may exploit various characteristics of the triplet $(\mathbf{s}, \mathbf{t}, \mathbf{a})$ at different levels (word, phrase, or sentence), and is derived from the representation of latent variables (or derivation) $\mathbf{a}$. In practice, the definition of latent variables, as well as the set of feature functions are two properties that characterize different approaches in Statistical Machine Translation.

Once the model is defined, its parameters are estimated on training corpora, traditionally relying on a two-step process. In the first step, several probabilistic models, which give values to feature functions f_1^M, are estimated independently on very large monolingual or bilingual corpora. The second step learns the mixing weights of these models to be used in the linear combination (1.3) via the *log-linear coefficients* $\boldsymbol{\lambda}$. The introduction of these parameters adds more flexibility, but also complexity. The optimization of $\boldsymbol{\lambda}$ forms the *tuning* task within the building of SMT systems. While the training of a standard maximum entropy model consists of maximizing the conditional likelihood, the training of log-linear coefficients within a SMT system often incorporates automatic evaluation metrics for MT (the most widely used is BLEU, see Section 2.5.2). A typical example is the *Minimum Error Rate Training* (MERT) (Och, 2003).

1.1.1 Word-based approach

The word-based approach is introduced in (Brown et al., 1993; Vogel et al., 1996; Tillmann et al., 1997) which propose to estimate the translation model (TM) $\mathbf{p}(\mathbf{s}|\mathbf{t})$ in the noisy channel model by decomposing the translation at the level of words, and by introducing word-to-word alignments as latent variables :

$$\mathbf{p}(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{a}} \mathbf{p}(\mathbf{s}, \mathbf{a}|\mathbf{t}) = \sum_{\mathbf{a}} \mathbf{p}(\mathbf{a}|\mathbf{t}) \times \mathbf{p}(\mathbf{s}|\mathbf{a}, \mathbf{t}) \tag{1.4}$$

where $\mathbf{a} = a_1, ..., a_J$ denotes the word alignment variable. Each element a_j for $1 \leq j \leq I$ takes value from $\{1, ..., J\}$; here J and I denote the number of words in the source and target sentences. The first term $\mathbf{p}(\mathbf{a}|\mathbf{t})$ is referred to as the distorsion model which characterizes the syntactic reordering of the source sentence, while $\mathbf{p}(\mathbf{s}|\mathbf{a}, \mathbf{t})$ provides a translation model given the reordering. The translation model defines the generative probability of $\mathbf{s}$ which can be obtained from word-level probabilities conditioning on aligned target

[1] Theoretically, $\mathbf{t}$ must be chosen to maximize its probability given $\mathbf{s}$, which is estimated by marginalizing over the variable $\mathbf{a}$ in Equation (1.2). In practice, as the marginalization is intractable, we instead look for a pair $(\mathbf{t}, \mathbf{a})$ that maximizes the probability, and take the corresponding $\mathbf{t}$ as the output translation.

words (lexical translation model), while the alignment $\mathbf{a}$ can be made independent from $\mathbf{t}$ as proposed in (Vogel et al., 1996) :

$$\begin{aligned}\mathbf{p}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \mathbf{p}(\mathbf{a}|\mathbf{t}) \times \mathbf{p}(\mathbf{s}|\mathbf{a},\mathbf{t}) \\ &= \left(\prod_{i=1}^{J} \mathbf{p}(a_i|a_{i-1})\right)\left(\prod_{i=1}^{J} \mathbf{p}(s_i|t_{a_i})\right)\end{aligned} \tag{1.5}$$

Other models propose different assumptions to make tractable the inference of latent variables $\mathbf{a}$. For example, IBM models (3 & 4) estimate the distorsion model $\mathbf{p}(\mathbf{a}|\mathbf{t})$ via the introduction of *fertility* that characterizes the number of source words a target word can align to. However, a common weakness of these models is the asymmetry of the alignment : a source word is aligned only to one target word, but a target word may align to multiple source words (many-to-one alignment). Moreover, most of them estimate $\mathbf{p}(\mathbf{s}|\mathbf{a},\mathbf{t})$ by simplifying it to a lexical translation model, which is a consequence of the many-to-one alignment : each target word t_{a_i} is generated depending only on the source word s_i it is aligned to. The model hence fails to capture information from target words other than the aligned word, or from a whole group of target words.

1.1.2 Phrase-based approach

The word-based approach has many shortcomings, such as the failing to capture dependencies outside the word alignment links, and to model complex reordering patterns. Moreover, longer dependencies are learnt only by the target language model which ignores the source text. The phrase-based approach aims at incorporating long contexts into the translation model by learning the translations for phrases. [2] The source sentence is first segmented into phrases, then each phrase is translated independently before they are recomposed to form the target sentence. However, compared to the word-based models, this kind of models introduces an additional level of complexity which is the phrase segmentation, while afterwards an optimal alignment is searched on phrases instead on words. To simplify the problem, the phrase segmentation is often performed using heuristic techniques during the training.

The shift from word-based to phrase-based models is introduced in (Och et al., 1999) with the method of alignment templates. This approach introduces latent phrase alignment variables directly derived from the word alignments, and which are constructed using a two-phase process : first, word-level alignments are built, then phrase-level alignments are derived from the first ones using an heuristic. To overcome the many-to-one mapping implied by word alignment models, the heuristic, often referred to as *symmetrization*, constructs two sets of word alignments, each in one direction, from the source to target side, and inversely. Then the procedure merges them into a symmetric alignment matrix that contains many-to-many alignments. Finally, for each sentence pair, all phrase pairs which are consistent with the alignment matrix are extracted. As in the word-based approach, the authors also define several generative models derived from aligned source and target phrases, which correspond to a set of feature functions in the discriminative framework.

[2] A phrase is simply defined as a sequence of words whose length is not fixed beforehand, and should not be understood as having here its usual linguistic meaning.

During training, these features take into account both the probability of a phrase pair, as well as the probabilities of inner lexical alignment links inside each phrase pair. The last terms will be later generalized and named *lexical weights* in (Koehn et al., 2003).

The alignment template method in (Och, 1999) considers bilingual word classes instead of surface forms as a first attempt to address data sparsity issues. Zens et al. (2002) introduce for the first time the term *phrase-based machine translation* which consists of a simplification of the alignment template approach where a distribution over bilingual phrases is estimated directly using their surface forms. On the level of sentences, a set of latent variables are introduced to describe the segmentation of a sentence pair into bilingual phrases. They also propose to deal with the reordering problem by the use of a reordering graph generated by a word-based model. The model described in (Koehn et al., 2003) is very similar, but introduces a simpler distance based reordering. These two last papers define the standard method for building a translation model from a parallel bilingual corpus.

Some propositions in the literature aim at directly addressing the joint segmentation and alignment into phrase pairs; however they are often based on complex models that need approximate inference procedures, or imply some constraints to restrict the combinatorial search space. The reader is invited to get further details in (Marcu and Wong, 2002; DeNero et al., 2006, 2008; Zhang et al., 2008; Andrés-Ferrer and Juan, 2009; Bansal et al., 2011; Feng and Cohn, 2013). However, these models never significantly outperform the standard methods described in (Zens et al., 2002; Koehn et al., 2003).

1.1.3 The n-gram approach in Machine Translation

In this section, we describe a variant of the phrase-based approach to Machine Translation that will be used in experiments in the rest of the dissertation. The n-gram based approach described here corresponds to LIMSI's in-house n-code implementation [3] (Crego et al., 2011) which has been exploited and showed to achieve state-of-the-art results in several translation evaluations, such as the WMT (Workshop on Statistical Machine Translation) and IWSLT (International Workshop on Spoken Language Translation) evaluation campaigns.

The n-gram-based approach to SMT borrows its main principle from the finite-state perspective (Casacuberta and Vidal, 2004) in which the translation process is divided in two sub-processes : a source reordering step and a (monotonic) translation step. The source reordering is based on a set of rewrite rules that non-deterministically reorder the input words so as to match the order of the target words (Crego and Mariño, 2006). The application of these rules yields a finite-state graph representing possible source reorderings, which are then used to carry out a monotonic translation from source to target phrases. The main latent variables in this approach is the segmentation of sentence pairs into elementary translation units called *tuples*, which are equivalent to phrase pairs in the phrase-based approach, and which correspond to pairs of variable-length sequences of source and target words. The n-gram-based approach hence differs from the standard phrase-based approach by the latent variables $\mathbf{a}$ that describe the reordering of the source

[3] ncode.limsi.fr/ .

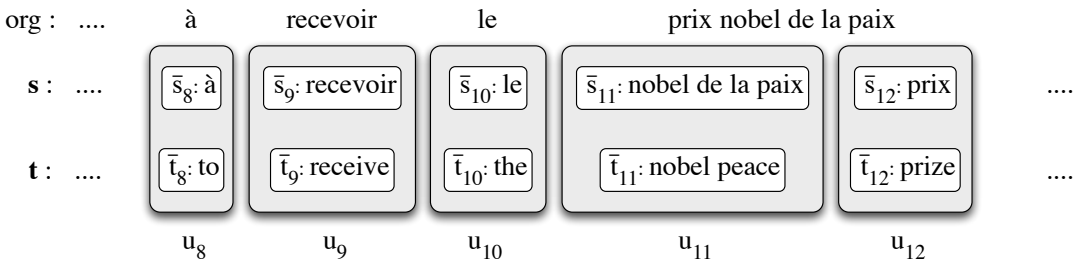


Figure 1.1 – The segmentation of a parallel phrase pair $(\mathbf{s}, \mathbf{t})$ into L bilingual *tuples* $(u_1, ..., u_L)$. The original source sentence (*org*) is shown above the reordered source sentence $\mathbf{s}$ and the target sentence $\mathbf{t}$. Each tuple u_i aligns a source phrase $\bar{s}_i$ to a target phrase $\bar{t}$.

sentence, and the joint segmentation into bilingual tuples. [4] Another particularity of the n-gram-based approach is the inclusion of n-gram translation models estimated on atomic tuples.

Feature functions in an n-gram-based SMT system

n-gram translation models (n-gram TMs) are estimated on the triplets $(\mathbf{s}, \mathbf{t}, \mathbf{a})$ based on a specific decomposition of the joint probability $\mathbf{p}(\mathbf{s}, \mathbf{t}|\mathbf{a})$. Given latent variables $\mathbf{a}$, the sentence pair can be regarded as a sequence of L bilingual units (or tuples) $(\mathbf{s}, \mathbf{t}) = u_1, ..., u_L$; each tuple u corresponds to a matching $u = (\bar{s}, \bar{t})$ between a source $\bar{s}$ and a target $\bar{t}$ phrase. Figure 1.1 gives an example of the joint reordering and segmentation of a sentence pair into tuples. Using the n-gram assumption at the level of tuples, the joint probability $\mathbf{p}(\mathbf{s}, \mathbf{t}|\mathbf{a})$ decomposes as :

$$\mathbf{p}(\mathbf{s}, \mathbf{t}|\mathbf{a}) = \mathbf{p}(u_1, ..., u_M) = \prod_{l=1}^{L} \mathbf{p}(u_l|u_{l-n+1}^{l-1})$$

which is in fact a bilingual n-gram model of tuples. This model can be estimated with techniques that will be described in Chapter 2, for instance using discrete probabilistic models with modified Kneser-Ney smoothing (Chen and Goodman, 1996), or by continuous-space model (Schwenk et al., 2007; Le et al., 2012a). The set of tuples is extracted during the training process from a word-aligned corpus (obtained for instance by MGIGA++ [5]) in such a way that a unique segmentation of the bilingual corpus is obtained.

In addition to the n-gram translation model, n-code also implements 10 feature functions that are very similar to those used in the standard phrase-based system :

- A target language model;
- Four *lexicon models*, two of them correspond to relative frequencies of tuples, and two lexical weights derived from word-aligned links inside each tuple (Koehn et al., 2003);

[4] The segmentation into phrases is deterministic, contrary to the reordering of the source sentence.
[5] http://www.kyloo.net/software/doku.php .

- Two *lexicalized reordering models* aiming at predicting the orientation of the next translation unit (Tillmann, 2004; Crego et al., 2011);
- A "weak" distance-based *distortion model* (Koehn et al., 2003);
- and finally a *word-bonus model* and a *tuple-bonus model* which compensate for the system preference for short translations.

Decoding

During the decoding, source sentences are represented in form of word lattices containing the most promising reordering hypotheses. They are derived from a set of rules extracted during the tuple extraction process, with the goal to reproduce reordering patterns observed in word alignments. Only those reorderings are used to process the translations. In practice, part-of-speech (POS) are used, instead of surface word forms, to increase the generalization power of these rules (Crego and Mariño, 2006).

1.1.4 Hierarchical phrase-based approach

The phrase-based approach has been shown to be powerful to learn and handle local reorderings and other translation decisions that are sensitive to local context, such as word insertions or deletions. However, long dependencies, such as those implied by structural linguistic constraints (agreement for instance) and the reordering of long phrases, are more difficult to handle, especially when only simple distance-based distortion models are used in such systems. Koehn et al. (2003) understate that in phrase-based approaches, phrases longer than 3 words hardly improve the performance, which highlights the implicit trade-off between the capacity of generating reorderings via phrases, and the difficulty related to their estimation. The hierarchical phrase-based approach (Chiang, 2005, 2007) results from the desire to capture longer translations whose scope is larger than a few consecutive words as in the standard phrases. The main idea is to use *hierarchical phrases*, which are phrases that contain sub-phrases, to learn the reorderings at the level of phrases. These hierarchical phrases are formally the productions of a synchronous context-free grammar (CFG) (Aho and Ullman, 1969), which is however learnt from training parallel corpora instead of being manually hand-crafted by linguist experts.

When expressed under the form of the log-linear model (Equation (1.3)), the hierarchical phrase-based approach hence only differs from the standard phrase-based approach by the representation of its feature functions, which is implemented as a weighted synchronous CFG where features are integrated into rule weights. The approach can be seen as an attempt to introduce some ideas of syntax-based SMT systems (Wu, 1997; Alshawi et al., 2000; Yamada and Knight, 2001), but to combine them with the strength of phrase-based SMT systems.

In the training step, the grammar rules are extracted from the training data and filtered out using heuristics in order to limit their number. The extraction process involves in a first step the phrase extraction as in the standard phrase-based approach, but then the

relative frequency of each phrase pair is equally distributed to all rules obtained from the extraction of this phrase pair into sub-phrases. The decoding is basically performed using CKY (Cocke-Kasami-Younger) algorithm for CFGs; the sentences are generated in a bottom-up fashion (instead of in left-to-right order as in phrase-based systems). This bottom-up decoding makes the integration of a LM more difficult. To do this, the n-gram LM is often viewed as a weighted finite state machine [6] which is then intersected with the weighted synchronous CFG of the hierarchical system; the LM scores will be part of the rule weights like other features (Chiang, 2007).

1.2 Evaluation in Statistical Machine Translation

Evaluating translations is challenging even in the presence of reference translations, and has been the central problem of an important amount of literature. An ideal evaluation may resort to the use of human assessments which, given a reference translation, can rank the adequacy of an output translation in terms of meaning and fluency of the output if used in the target language (White et al., 1994). However, collecting human judgements on translations is very expensive, hence is not feasible in an iterative system development when we need to routinely perform evaluations to identify if changes in the system are really beneficial (Lopez, 2008). A feasible alternative is automatic metrics, of which a remarkable number have been proposed in the literature, such as the *Word Error Rate* (WER) or WER-related metrics (Och et al., 1999), the *Bilingual Evaluation Understudy* (BLEU) (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), or the *Translation Edit Rate* (TER) (Snover et al., 2006). In the context of proliferating quality metrics, an important research direction is to identify the correlation of each metrics with the human judgement (Papineni et al., 2002; Callison-Burch et al., 2007). The metric evaluation task has been included in the WMT evaluation campaigns since 2008 [7], resulting in some interesting automatic metrics (see, for instance (Fishel et al., 2012; Lo and Wu, 2013; Joty et al., 2014; Stanojević and Sima'an, 2014)).

Despite its well-known weaknesses (Turian et al., 2006; Callison-Burch et al., 2006), BLEU is still the most widely used metric, and measures the similarity of n-gram count vectors between hypotheses and the reference translations. Let $\sharp(g, \mathbf{t})$ be the count of an n-gram g in a particular sentence $\mathbf{t}$ and n-grams($\mathbf{h}$) the set of different n-grams in a hypothesis $\mathbf{h}$, the n-gram precision p_n for a set of translations H and the set of their respective references R is computed as :

$$p_n(H, R) = \frac{\sum_{\mathbf{h} \in H} \sum_{g \in \text{n-grams}(\mathbf{h})} \sharp(g, \text{ref}(\mathbf{h}))}{\sum_{\mathbf{h} \in H} \sum_{g' \in \text{n-grams}(\mathbf{h})} \sharp(g', \mathbf{h})}$$

BLEU is a precision-based score, and is computed as the geometric average of multiples p_n up to a maximum value of n (4 in practice). The metric also includes a *brevity penalty* (BP) to penalize hypotheses with are much shorter than the references. The BP is a corpus-level score which compares the overall number of words in H (denoted by $|H|$)

[6] each state corresponds to a sequence of $n-1$ words.

[7] http://www.statmt.org/wmt15/metrics-task/ for the 2015 edition.

with the one of the reference set $|R|$:

$$BP(H,R) = \begin{cases} 1 & \text{if } |H| > |R| \\ \exp\left(1 - \frac{|R|}{|H|}\right) & \text{otherwise} \end{cases}$$

and $\text{BLEU}(H,R) = BP(H,R) \times \exp\left(\sum_{n=1}^{4} \log p_n\right)$.

In this dissertation, we always use BLEU as the metric to evaluate the performance of a SMT system, hence the performance of CSMs when they are used for SMT applications. Our practice follows the claim in (Callison-Burch et al., 2006) that "the use of BLEU is recommended to be restricted to tracking, broad, *incremental changes* to a single system, or comparing systems which employ similar translation strategies". The integration of continuous-space models into SMT systems is a perfect example of such incremental changes, while we will always compare only between CSMs included in the same baseline. Moreover, BLEU is a corpus-level score but has some sentence-level approximations (Lin and Och, 2004; Liang et al., 2006; He and Deng, 2012) which facilitate its incorporation into discriminative training procedures that will be described in the next section.

1.3 Discriminative tuning and training of SMT systems

In this section, we focus on the discriminative framework for Statistical Machine Translation (Equation (1.3)), especially on some tuning algorithms which are used specifically to optimize the log-linear coefficients $\boldsymbol{\lambda}$, and on training procedures aiming at jointly optimize multiple models according to a well-understood objective function. Ideas from this section can be straightforwardly applied to the training of continuous-space models, as described later in Chapter 5.

The standard building of the whole log-linear model relies on a two-step process. First, several probabilistic models are estimated independently on very large monolingual or bilingual corpora, and output the parameters of the feature functions $\{f_i, 1 \leq i \leq M\}$. Then, an additional tuning step is needed to balance the contribution of each model, resulting in *global model weights* (i.e log-linear coefficients) optimized based on the overall translation quality. These two steps are called respectively *training* step (for the feature functions), and *tuning* step (for $\boldsymbol{\lambda}$). While the training step uses training (monolingual and parallel) data, the tuning step is performed typically on a separate development data, using for instance *Minimum Error Rate Training* (MERT) (Och, 2003).

Another approach aiming at globally optimized both groups of parameters for SMT systems has been proposed in (Liang et al., 2006; Blunsom et al., 2008; Blunsom and Osborne, 2008; Dyer and Resnik, 2010; Lavergne et al., 2011, 2013). In this framework, all these parameters are learnt discriminatively in a unified manner, using an optimization method on a well-understood objective function over the entire training set. This framework dispenses the need to build separate models and to tune their interpolation weights; indeed these two steps are performed simultaneously.

In practice, both approaches suffer from the richness of feature sets which become

larger to incorporate richer linguistic knowledge. In the literature, solutions have been proposed to overcome this challenge, firstly by making the tuning step more robust in the presence of a large number of feature functions. For example, Tillmann and Zhang (2006) propose to use structured SVMs. Another trend explores various online learning approaches : structured perceptron (Liang et al., 2006; Wisniewski and Yvon, 2013), *Margin Infused Relaxed Algorithm* (MIRA) (Watanabe et al., 2007; Chiang et al., 2008), learning as ranking (Hopkins and May, 2011). Much of these papers however focus on the tuning step (on small development sets) while leaving the training procedure (on large training sets) unchanged. There are some attempts to address the training on large datasets of component models (Liang et al., 2006; Blunsom et al., 2008; Dyer and Resnik, 2010). These discriminative SMT systems will be described in the second part of the section (1.3.2), while the first part (1.3.1) is dedicated to a review of classical tuning algorithms.

1.3.1 Discriminative tuning algorithms

We start this section by a formal description adopted from (Cherry and Foster, 2012) of the tuning problem. The training data of this step is often called *development set*. Suppose the development set $\mathcal{D}_{dev}$ contains d source sentences $\{\mathbf{s}\}_{i=1}^{d}$. For each source sentence $\mathbf{s}_i$, let $\mathbf{R}_i$ be the set of target reference sentences, in which there may be more than one reference. Finally, $\mathbf{H}_i$ designs a subset of reachable hypotheses generated by the decoder; each $\mathbf{h} = (\mathbf{t}, \mathbf{a}) \in \mathbf{H}_i$ is composed of a target sentence $\mathbf{t}$ and a derivation $\mathbf{a}$ which represents the steps by which the target sentence is generated from the source; $\mathbf{a}$ contains latent variables that are system-dependent (Section 1.1). The search space depends on the decoder; in practice it is computationally expensive to enumerate all hypotheses in this space. Henceforth, we often work with a representation of $\mathbf{H}_i$, as an N-best list, or as a lattice of hypotheses. Depending on the context, the notation $\mathbf{H}_i$ may be used, abusively, to denote either of these representations.

Hypotheses from $\mathbf{H}_i, i = 1, ..., d$ are scored by the following linear function :

$$F_{\boldsymbol{\lambda}}(\mathbf{h}_j) = \sum_{m=1}^{M} \lambda_m f_m(\mathbf{h}_j) = \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}_j) \tag{1.6}$$

where $\mathbf{f}(\mathbf{h}_j)$ denotes the M-dimensional vector containing all $\mathbf{h}_j$'s features. The log-linear model produces 1-best output according to this scoring function, $\hat{\mathbf{h}}_i = \underset{\mathbf{h}_j \in \mathbf{H}_i}{\operatorname{argmax}} \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}_j)$. Finally, all 1-best outputs corresponding to the source sentences $\mathbf{s}_{i=1}^{d}$ are assembled and the whole document is evaluated by an automatic quality metric (typically BLEU score (Papineni et al., 2002)) computed using the references $\mathbf{R}_{i=1}^{d}$ (see Section 2.5.2 for more details on MT evaluation). The log-linear coefficients $\boldsymbol{\lambda}$ are learnt so that the assembled document is best evaluated using the BLEU score. N-best lists or lattices are often used to approximate the search space of reachable hypotheses. In the traditional MERT infrastructure, an outer loop is maintained to iterate between the optimization of $\boldsymbol{\lambda}$ and the re-decoding along with those new values to enhance the approximation by N-best lists (Och, 2003) or by lattices(Macherey et al., 2008).

Algorithm 1 A generic batch tuning algorithm

1: Init. $\boldsymbol{\lambda}$
2: **for** each iteration **do** do
3: Update $\mathbf{H}_1^d$ using current values of $\boldsymbol{\lambda}$
4: Train $\boldsymbol{\lambda}$ to optimize the overall criterion (1.7)
5: **end for**
6: Return $\boldsymbol{\lambda}$

Training procedures

Almost all tuning algorithms (except MIRA) directly optimize an objective function defined on the development set $\mathcal{D}_{dev} = \{\mathbf{s}_{i=1}^d\}$ and the approximate search spaces $\mathbf{H}_{i=1}^d$. Since the used MT metrics (such as BLEU) are document-level scores, the optimized criteria could be a surrogate document-level function in order to follow more closely the system performance on $\mathcal{D}_{dev}$. However, to simplify the training procedures, linear criteria are often used which decompose over training examples :

$$\mathcal{L}(\boldsymbol{\lambda}, \mathcal{D}_{dev}) = \sum_{i=1}^{d} \mathcal{L}(\boldsymbol{\lambda}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i) + \mathcal{R}(\boldsymbol{\lambda}) \tag{1.7}$$

where $\mathcal{R}(\boldsymbol{\lambda})$ is a regularization term, $\mathcal{R}(\boldsymbol{\lambda}) = \gamma \times \frac{||\boldsymbol{\lambda}||^2}{2}$. Each decomposed term $\mathcal{L}(\boldsymbol{\lambda}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i)$ depends on the source $\mathbf{s}_i$, the set of references $\mathbf{R}_i$ as well as on a (approximate) search space $\mathbf{H}_i$. This space can be updated as soon as the optimizer provides new values for $\boldsymbol{\lambda}$. However, in theoretical as well as practical terms, it would be much simpler to assume fixed $\mathbf{H}_{i=1}^d$, i.e to limit the tuning procedure inside lines 3 and 4 in Algorithm 1. Outside this scope, the optimizer can be viewed as being a part of an outer loop that alternatively optimizes $\boldsymbol{\lambda}$ and re-decodes the search space. Algorithm 1 is also the general architecture of MERT (Och, 2003) and other related procedures that we often refer to as *batch* algorithms. Within this framework, tuning algorithms differ in their criteria and how they optimize the parameters given each criterion.

On the other hand, *online* algorithms only consider one source sentence $\mathbf{s}_i$ at a time and update $\boldsymbol{\lambda}$ according to a criterion $\mathcal{L}(\boldsymbol{\lambda}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i)$, where $\mathbf{H}_i$ may be time-dependent. At each iteration, these updates aim at guiding the decoder to output 1-best translations that are closest to elements of $\mathbf{R}_i$. However, a decoder usually cannot generate translations that match exactly the reference translations due to beam search pruning and out-of-vocabulary (OOV) words, henceforth the criterion can not be computed solely based on $\mathbf{R}_i$. A solution is to use a set of oracle translations (Tillmann and Zhang, 2006; Liang et al., 2006; Watanabe et al., 2007), chosen from $\mathbf{H}_i$ and $\mathbf{R}_i$: $\mathcal{O}_i = \mathcal{O}(\mathbf{H}_i, \mathbf{R}_i)$, and to update $\boldsymbol{\lambda}$ towards these hypotheses. These are reachable approximations of the translations in $\mathbf{R}_i$ according to a distance derived from a quality metric, such as BLEU.

Updating $\boldsymbol{\lambda}$ using MIRA (proposed in (Crammer et al., 2006), applied to SMT in (Watanabe et al., 2007; Chiang et al., 2008, 2009; Cherry and Foster, 2012)) is a notable exception that does not explicitly use an objective function. However, its original version can be shown to be the Lagrange dual form of a structured hinge loss [8] optimized

[8] which is also named *max-margin* criterion in this dissertation.

by coordinate ascent (Martins et al., 2010). Later variants of MIRA in (Chiang et al., 2008) modify the original update steps to better suit reachable spaces used in SMT, but are not derived from any mathematically defined criterion. An interpretable version of MIRA is provided in (Gimpel and Smith, 2012).

Minimum Error Rate and Minimum Risk training

Minimum Error Rate Training (MERT (Och, 2003)) is the first proposal to train a log-linear model by incorporating a SMT evaluation metric. Assume that for each hypothesis translation $\mathbf{h}$ from a source sentence $\mathbf{s}_i \in \mathcal{D}_{dev}$, an error score $E_i(\mathbf{h})$ can be obtained by comparing $\mathbf{h}$ with elements in $\mathbf{R}_i$. The number of errors counted on $\mathcal{D}_{dev}$ is defined to be the sum of errors of each individual sentence, exactly like the decomposition into sentence-level losses of Equation (1.7) :

$$\mathcal{L}_{mert}(\boldsymbol{\lambda}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i) = E_i(\hat{\mathbf{h}}_i) \tag{1.8}$$

where $\hat{\mathbf{h}}_i$ is the hypothesis that maximizes the model score :

$$\hat{\mathbf{h}}_i = \underset{\mathbf{h} \in \mathbf{H}_i}{\operatorname{argmax}} \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}) \tag{1.9}$$

However, the criterion (1.8) is not easy to be optimized, as it contains an *argmax* operation (1.9) which makes the sub-gradient computation difficult; this criterion is indeed not optimized using gradient-based methods. Instead, the objective function can be optimized using Powell's algorithm, combined with a grid-based line optimization method (Och, 2003). The optimization starts from a random initial value of $\boldsymbol{\lambda}$, and tries to find a better value by optimizing one dimension of $\boldsymbol{\lambda}$ while keeping the others fixed. This procedure has a weakness that it suffers from a lot of different local optima of the objective function. In order to overcome this difficulty, one often runs the algorithm from various random initial points.

Even if this issue with local optima can somehow be mitigated, this algorithm is still slow as it considers a line optimization for each dimension of $\boldsymbol{\lambda}$ at each iteration. Och (2003) propose an improved version of the grid-based line optimization method, which is much faster and more stable than the original one. In order to optimize the BLEU score, the error function E_i needs to be refined so that its document-level accumulation approximates the metric. However, the algorithm does not scale with a large-dimension $\boldsymbol{\lambda}$ representing the case where we have a lot feature functions. Indeed, MERT is shown in practice to only work well up to 30 features (Chiang et al., 2008; Cherry and Foster, 2012). Some attempts have been made to improve this line optimization procedure (Smith and Eisner, 2006; Cer et al., 2008). Moore and Quirk (2008) propose several ways of generating random starting points for MERT, whereas Cer et al. (2008) come with a modified version of Powell's algorithm in which diagonal directions are chosen at random. Macherey et al. (2008) apply the MERT search on lattices of hypotheses instead of on N-best lists, which results in a smoother convergence. The proposals in (Foster and Kuhn, 2009) aim to improve the stability of the tuning procedure. However, in order to scale, it is likely that one should replace the grid-based search in favour of simpler and more stable optimization methods, such as Stochastic Gradient Descent (SGD). Nevertheless,

the ability of MERT to optimize parameters according to an evaluation metric (BLEU) is an advantage compared to the standard maximum-likelihood training, and has become a standard assumption in SMT.

An application of SGD to MERT may be possible by replacing the piecewise linear objective function (1.8) by a smoothed error count criterion :

$$\mathcal{L}_{MR}(\boldsymbol{\lambda}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i) = \sum_{\mathbf{h} \in \mathbf{H}_i} E_i(\mathbf{h}) \mathbf{p}(\mathbf{h}|\mathbf{s}_i) \tag{1.10}$$

where, instead of considering only the hypothesis maximizing the model score, the criterion accounts for an expected error within each search space $\mathbf{H}_i$. This expectation is estimated based on a probabilistic interpretation of the log-linear model similar to Equation (1.2) that we rewrite here as follows :

$$\mathbf{p}(\mathbf{h}'|\mathbf{s}_i) = \frac{\exp(\alpha \times \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}'))}{\sum\limits_{\mathbf{h} \in \mathbf{H}_i} \exp(\alpha \times \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}))} \tag{1.11}$$

where α is a scaling factor. Compared to the original un-smoothed function, this smoothed error function has fewer local optima with much more stable optimization results. The training with this loss is often referred to as *Minimum Risk* training, and is commonly used in Automatic Speech Recognition community (Juang et al., 1996; Schlüter and Ney, 2001). Within SMT community, the error $E_i(.)$ can be assigned to the negative sentence-BLEU (Watanabe et al., 2007; Chiang et al., 2008), hence the procedure amounts to an expected BLEU training (Zens et al., 2007; Rosti et al., 2011; He and Deng, 2012).

Max-margin approach

Assuming that $\mathbf{H}_i$ represents the space of all derivations that can be obtained from the source sentence $\mathbf{s}_i$, max-margin methods minimize (by the batch algorithm) the hinge loss, or structured perceptron (Collins, 2002; Taskar et al., 2004) :

$$\mathcal{L}_{max-margin}(\boldsymbol{\lambda}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i) = \max_{\mathbf{h} \in \mathbf{H}_i} \left[\text{cost}_i(\mathbf{h}) + \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}) - \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}^{*i}) \right] \tag{1.12}$$

where $\mathbf{h}^{*i}$ is the oracle hypothesis :

$$\mathbf{h}^{*i} = \operatorname*{argmax}_{\mathbf{h} \in \mathbf{H}_i} \text{BLEU}_i(\mathbf{h})$$

and the cost function $\text{cost}_i(.)$ computes the difference in BLEU, $\text{cost}_i(\mathbf{h}) = \text{BLEU}_i(\mathbf{h}^{*i}) - \text{BLEU}_i(\mathbf{h})$. The same hinge loss is optimized by MIRA, however by an online training procedure. The definition of this criterion implies the use of an oracle $\mathbf{h}^{*i}$ and another hypothesis from $\mathbf{H}_i$ involved in the *max* operation, similarly to the local updating towards a "hope" and against a "fear" derivation in (Liang et al., 2006). In practice, $\mathbf{h}^{*i}$ can vary during the training, by first choosing a set of oracle translations $\mathcal{O}_i$ which contains k best hypotheses from $\mathbf{H}_i$, then at each iteration, $\mathbf{h}^{*i}$ is chosen from $\mathcal{O}_i$ which maximizes the model score (with the current $\boldsymbol{\lambda}$ value). This procedure makes the local update even more local and conservative. The criterion becomes zero only if the model score of $\mathbf{h}^{*i}$, is higher than the model score of each of the other hypotheses up to a positive margin, defined based on the BLEU distance.

Pairwise ranking

Instead of considering only one pair of hope and fear hypotheses as does the max-margin criterion (1.12), *Pairwise Ranking Optimization* (PRO) considers a set of critical pairs $\mathcal{C}_i = \{(\mathbf{h}_g, \mathbf{h}_b), \text{BLEU}_i(\mathbf{h}_g) > \text{BLEU}_i(\mathbf{h}_b)\}$, then uses a binary classifier (SVM for instance) to separate between pairs in which hypotheses are correctly (according to a MT metric) ranked by the model score (i.e $\boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}_g) > \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}_b)$), from other pairs. Introduced for the first time in (Hopkins and May, 2011), the corresponding objective function used by SVM rank can be written as a sum of structured perceptrons :

$$\mathcal{L}_{PRO-1}(\boldsymbol{\lambda}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i) = \sum_{(\mathbf{h}_g, \mathbf{h}_b) \in \mathcal{C}_i} \max\{0, 1 + \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}_b) - \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}_g)\} \tag{1.13}$$

which differs from the max-margin approach and MIRA (described later) by the replacement of the *max* operator by a sum over critical pairs, and by the margin fixed to 1. This sum of hinge losses is 0 only if each pair in $\mathcal{C}_i$ is separated by model scores up to 1. It is however possible to replace the fixed margin by a general *cost-augmented* score as in Equation (1.12) :

$$\mathcal{L}_{PRO}(\boldsymbol{\lambda}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i) = \sum_{(\mathbf{h}_g, \mathbf{h}_b) \in \mathcal{C}_i} \max\{0, \text{cost}_i(\mathbf{h}_b) - \text{cost}_i(\mathbf{h}_g) + \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}_b) - \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}_g)\} \tag{1.14}$$

The cost function $\text{cost}_i(\mathbf{h})$ represents the cost paid by choosing the hypothesis $\mathbf{h}$ instead of an oracle (which is fixed beforehand). This cost can be defined in terms of sentence-level BLEU distance as in the max-margin approach; because $\text{BLEU}_i(\mathbf{h}_g) > \text{BLEU}_i(\mathbf{h}_b)$, we always have positive margins $\text{cost}_i(\mathbf{h}_b) - \text{cost}_i(\mathbf{h}_g) > 0$. PRO criterion (Equation (1.13)) uses a sum over pairs, so does not need a hope-and-fear derivation selection. All pairs, once considered critical, are treated equally by a black-box binary classifier or optimizer. This may be a weakness of PRO that all pairs, which are correctly ranked, are however not equally important to have a high BLEU score. On the other hand, using multiple pairs may help to feed a large amount of information from $\mathbf{H}_{i=1}^{d}$ to the classifier, hence prevent over-fitting, especially when there is a large amount of free parameters to be trained. We will discuss this issue later in Chapters 5.

In PRO as described in (Hopkins and May, 2011), $\mathcal{C}_i$ is selected by a sampling routine which prefers pairs with large sentence-BLEU (Watanabe et al., 2007; Chiang et al., 2008) differences. The need to such sampling is due to the fixed margin used in the original form (1.13). Indeed, it would be unreasonable to require all critical pairs to be separated similarly by the same margin no matter how the two hypotheses resemble; the sampling procedure is hence useful in that it forces the optimizer to use more intensively pairs with important BLEU differences. In the second form (1.14), the margin is adjusted according to the distance inside each pair, hence such sampling scenario may be unnecessary.

ListNet training

Like the Pairwise Ranking, tuning with ListNet algorithm, first proposed in (Niehues et al., 2015), also considers the tuning problem as a ranking problem on N-best lists;

the procedure does not aims only at reducing the gap between the best BLEU score and the best model score hypotheses, but regards the ranking of all hypotheses as a whole. The ListNet algorithm (Cao et al., 2007) is a list-wise approach to the ranking problem. The main idea is to define two probability distributions respectively on two rankings, one based on model scores, and another based on an external evaluation metric (like BLEU, or more precisely sentence-level BLEU).

Given two distributions $\mathbf{p}_1$ and $\mathbf{p}_2$ defined on $x \in \mathcal{X}$, their divergence can be expressed by the cross-entropy :

$$\mathcal{L}_{cross-entropy}(\mathbf{p}_1, \mathbf{p}_2) = -\sum_{x \in \mathcal{X}} \mathbf{p}_1(x) \log(\mathbf{p}_2(x)) \tag{1.15}$$

The model score-based distribution is being learnt to follow the one based on BLEU score; the criterion is defined in terms of the divergence between these two distributions, expressed by the above formulation. For our tuning problem, the two distributions are defined over $\mathbf{h} \in \mathbf{H}_i$: $\mathbf{p}_2(.)$ is defined in the framework of the log-linear model (1.11) :

$$\mathbf{p}_{\boldsymbol{\lambda}}(\mathbf{h}'|\mathbf{s}_i) = \frac{\exp(\beta \times \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}'))}{\sum\limits_{\mathbf{h} \in \mathbf{H}_i} \exp(\beta \times \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}))}$$

which depends on current values of $\boldsymbol{\lambda}$, and $\mathbf{p}_1(.)$ is derived from sentence-BLEU scores :

$$\mathbf{p}_{\mathrm{BLEU}_i}(\mathbf{h}') := \mathbf{p}_{\mathrm{BLEU}}(\mathbf{h}'|\mathbf{s}_i) = \frac{\exp(\gamma \times \mathrm{BLEU}_i(\mathbf{h}'))}{\sum\limits_{\mathbf{h} \in \mathbf{H}_i} \exp(\gamma \times \mathrm{BLEU}_i(\mathbf{h}))}$$

The two hyper-parameters β and γ are used to address the situation where the distribution provides a strong probability (almost 1) to only one hypothesis, and near-zeros probabilities to others. Henceforth, the ListNet criterion, which still match the general form of the objection function (1.7), is :

$$\begin{aligned}\mathcal{L}_{ListNet}(\boldsymbol{\lambda}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i) &= -\sum_{\mathbf{h} \in \mathbf{H}_i} \mathbf{p}_{\mathrm{BLEU}_i}(\mathbf{h}) \times \log(\mathbf{p}_{\boldsymbol{\lambda}}(\mathbf{h}|\mathbf{s}_i)) \\ &= -\sum_{\mathbf{h} \in \mathbf{H}_i} \mathbf{p}_{\mathrm{BLEU}_i}(\mathbf{h}) \times (\beta \times \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}) - \log \mathrm{H}_{\boldsymbol{\lambda}}(\mathbf{s}_i, \mathbf{H}_i))\end{aligned} \tag{1.16}$$

where $\mathrm{H}_{\boldsymbol{\lambda}}(\mathbf{s}_i, \mathbf{H}_i)$ is the normalization constant, $\mathrm{H}_{\boldsymbol{\lambda}}(\mathbf{s}_i, \mathbf{H}_i) = \sum\limits_{\mathbf{h}' \in \mathbf{H}_i} \exp(\beta \times \boldsymbol{\lambda}.\mathbf{f}(\mathbf{h}'))$.

This criterion can be optimized using SGD. Unlike the structured perceptron and PRO, but like the expected-BLEU (or Minimum Risk) training, its gradient involves all hypotheses in $\mathbf{H}_i$. The criterion also does not require the definition of oracles or "fear" derivations. However, the representation of $\mathbf{H}_i$ is required to be an N-best list, otherwise it would be intractable to estimate the two probability distributions defined above.

Online learning methods

The *batch* regime (Algorithm 1) defines a training criterion over all source sentences $\mathbf{s}_{i=1}^{d}$. Online learning methods, at the other end, visit one training example $\{\mathbf{s}_i, \mathbf{R}_i\}$ at each

Algorithm 2 A generic online tuning algorithm

1: Training data $\mathcal{D} = \{\mathbf{s}_i, \mathbf{R}_i\}_{i=1}^{d}$
2: Init. search space $\mathbf{H}_i = \{\}, i = 1, ..., d$
3: Init. oracles $\mathcal{O}_i = \{\}, i = 1, ..., d$
4: Init. $\boldsymbol{\lambda}^0$
5: **for** each iteration **do** do
6: **for** $i = 1, ..., d$ **do** do
7: Update $\mathbf{H}_i$ with $\boldsymbol{\lambda}^{i-1}$ (also called *fear decoding*)
8: Update oracles $\mathcal{O}_i = \mathcal{O}(\mathbf{R}_i, \mathbf{H}_i)$ (*hope decoding*)
9: Search for $\boldsymbol{\lambda}^i = \underset{\boldsymbol{\lambda}}{\operatorname{argmin}} \sum_{k=1}^{i} \mathcal{L}(\boldsymbol{\lambda}, \mathbf{s}_k, \mathbf{R}_k, \mathbf{H}_k) + \mathcal{R}(\boldsymbol{\lambda})$, eventually from the previous iteration's value $\boldsymbol{\lambda}^{i-1}$
10: **end for**
11: **end for**
12: Return $\frac{1}{d} \sum_{i=1}^{d} \boldsymbol{\lambda}^i$ (we suppose one-pass algorithm)

iteration, but proceed to the re-decoding to have a new $\mathbf{H}_i$. Algorithm 2 outlines a generic view on this kind of learning. Online algorithms differ in instructions at lines 6, 7 and 8, i.e the approximation to the search space $\mathbf{H}_i$, the update of oracle translation sets, the objective function as well as the algorithm used to update $\boldsymbol{\lambda}$.

Intuitively, the search space can be approximated by decoding an N-best list containing the N reachable hypotheses that are best evaluated by the scoring function (1.6), using the current value $\boldsymbol{\lambda}^{i-1}$. From $\mathbf{H}_i$, oracles are obtained by choosing hypotheses that are best evaluated by (sentence-)BLEU score. The oracle set satisfies henceforth two conditions : these are good translations according to the MT quality metric, but still reachable by the current decoder. In practice, the derivation of these candidates varies strongly from one method to another. In (Tillmann and Zhang, 2006), oracle translations are searched as follows : a regular phrase-based decoder is modified in a way that it uses BLEU scores as the optimization criterion independent of any translation model; for each source sentence $\mathbf{s}_i$, a top five hypotheses are computed in a preliminary phase, and are stored separately before the training starts. The authors also introduce the use of *relevant sets* to replace the role played by N-best lists in approximating the search space. These sets include only hypotheses which are the most easy to be confused by the current scoring function with one of the oracle translations; here the distance between hypotheses is based on the definition of a convex cost-sensitive margin. The main assumption made by this procedure is that if $\boldsymbol{\lambda}$ is trained to well distinguish oracles from their most closest alternatives, then it would work well on the whole search space. According to this, no N-best list is needed during the training; the decoder is modified, at each iteration, and for each of the five oracle translation, to decode the hypothesis which is scored the most similarly to this oracle. Compared to the general algorithm 2, only relevant sets are computed at line 6, while the oracles (line 7) are searched and stored before any training starts [9]. This procedure, however, does not fundamentally differs from the general scenario, as the resulting update of $\boldsymbol{\lambda}$ is proved not to be affected by the replacement of $\mathbf{H}_i$ in favour of the relevant sets.

[9] Moreover, relevant sets depend directly on these oracles.

Liang et al. (2006) propose to compare two methods of defining and updating towards oracles :

- *Bold updating:* Update towards the highest scoring hypothesis $\mathbf{h}_j = (\mathbf{t}_j, \mathbf{a}_j)$, where $\mathbf{t}_j$ is constrained to be one of the reference translations, $\mathbf{t}_j \in \mathbf{R}_i$, but $\mathbf{a}_j$ is unconstrained. Examples not reachable by the decoder are skipped.

- *Local updating:* Update towards an oracle chosen from an N-best list ($\mathbf{H}_i$) which achieves the highest BLEU score.

Both updating scenarios are similar to the perceptron algorithm. *Bold updating* resembles the traditional perceptron update rule where parameters are updated towards a unique annotated label. *Local updating* only uses hypotheses from $\mathbf{H}_i$, similarly to the pairwise ranking algorithm. Experiments in (Liang et al., 2006) show that the bold updating appears to over-fit severely, while the local updating seems to be much more stable. Recently, this point of view has been challenged in (Yu et al., 2013). Wisniewski and Yvon (2013) describe a similar sub-gradient training, but involve a cost function based on a linear combination of the n-gram precisions accounted in the BLEU score (Section 1.2).

Margin Infused Relaxed Algorithm : An update from $\boldsymbol{\lambda}^{i-1}$ to $\boldsymbol{\lambda}^{i}$ can be defined through the framework of SGD, by computing the gradient of $\mathcal{L}(., \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i)$ at $\boldsymbol{\lambda}^{i-1}$ (Tillmann and Zhang, 2006; Liang et al., 2006; Wisniewski and Yvon, 2013). Its final update is quite similar to the *batch-version* of (1.7). Another promising update method is used by MIRA (Crammer et al., 2006), which is an online version of the max-margin approach for structured classification problem (Collins, 2002; Taskar et al., 2004). The original MIRA employs a hinge loss with a cost function corresponding to BLEU-based difference :

$$\mathcal{L}_{mira}(\boldsymbol{\lambda}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i) = \max_{\mathbf{h}^* \in \mathcal{O}_i, \mathbf{h} \in \mathbf{H}_i} [\text{BLEU}_i(\mathbf{h}^*) - \text{BLEU}_i(\mathbf{h}) + \boldsymbol{\lambda}.(\mathbf{f}(\mathbf{h}) - \mathbf{f}(\mathbf{h}^*))] \tag{1.17}$$

where the max is taken over all pairs $\{\mathbf{h}, \mathbf{h}^*\}$ such that $\mathbf{h}$ is a reachable hypothesis, while $\mathbf{h}^*$ is an oracle translation. $\text{BLEU}_i(\mathbf{h})$ may be a sentence-BLEU score computed according to reference translations in $\mathbf{R}_i$ (Watanabe et al., 2007), or an approximated *pseudo* corpus-level BLEU in which the reference translation of $\mathbf{s}_i$ is replaced by $\mathbf{h}$ (Chiang et al., 2008). The criterion is 0 only if $\boldsymbol{\lambda}$ separates each $\mathbf{h} \in \mathbf{H}_i$ from each $\mathbf{h}^* \in \mathcal{O}_i$ by a margin which is their BLEU differences.

According to (Martins et al., 2010), the problem at line 9 of Algorithm 2 can be re-written, using the Lagrange dual form as :

$$\boldsymbol{\lambda}^i = \underset{\boldsymbol{\lambda}}{\text{argmin}} \frac{\gamma}{2} ||\boldsymbol{\lambda} - \boldsymbol{\lambda}^{i-1}||^2 + \mathcal{L}(\boldsymbol{\lambda}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i)$$

The interpretation is that, the new value $\boldsymbol{\lambda}^i$ should optimize the newly added term $\mathcal{L}(., \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i)$, by making the smallest change to $\boldsymbol{\lambda}^{i-1}$. The online learning in (Crammer et al., 2006) interprets lines 7 and 8 by the search for a hypothesis $\mathbf{h}_j$ and an oracle $\mathbf{h}^*$ which are the maximizers of (1.17). Let $\{\mathbf{h}_j, \mathbf{h}^*\}$ be this pair, the update can be performed

in two steps :

$$\eta_i = \min\{C, \frac{\mathcal{L}(\boldsymbol{\lambda}^{i-1}, \mathbf{s}_i, \mathbf{R}_i, \mathbf{H}_i)}{||\mathbf{f}(\mathbf{h}_j) - \mathbf{f}(\mathbf{h}^*)||^2}\}$$
$$\boldsymbol{\lambda}^i = \boldsymbol{\lambda}^{i-1} + \eta_i(\mathbf{f}(\mathbf{h}^*) - \mathbf{f}(\mathbf{h}_j)) \quad (1.18)$$

In (Watanabe et al., 2007), the oracle translation are updated at each iteration (line 8), while BLEU scores are a sentence-level approximation of the document-level score. In (Chiang et al., 2008), the BLEU scores are estimated on a *pseudo* document containing all best hypotheses (according to the scoring function) from previous updates. The set of oracles translation in that work is modified to a single *hope* derivation which maximizes $\boldsymbol{\lambda}.\mathbf{f}(.) + \text{BLEU}_i(.)$, whereas the *fear* derivation is the maximizer of (1.17) given the *hope* oracle, i.e the maximizer of $\boldsymbol{\lambda}.\mathbf{f}(.) - \text{BLEU}_i(.)$. The search for these two hypotheses is performed at each iteration by calling two modified 1-best decoders; the use of N-best lists is no longer needed. A detailed description of MIRA can be found in (Cherry and Foster, 2012).

Batch MIRA

Compared to batch algorithms (Algorithm 1), online learning (Algorithm 2) requires to decode within the loop, which is not really convenient. For instance, the decoder updates the approximate search space $\mathbf{H}_i$ at each iteration, which prevents us to store all hypotheses at once in memory. Yet it is still possible to benefit from MIRA's update steps within a batch algorithm. Cherry and Foster (2012) propose a batch version of the online MIRA algorithm, which intervenes in line 4 of Algorithm 1.

Batch MIRA differs from its online version only by the composition of hope and fear hypotheses; they are estimated on the N-best lists (or lattices) obtained with the precedent value $\boldsymbol{\lambda}^{i-1}$. Depending on the representation of the approximate search space, two versions of batch MIRA have been proposed : batch N-best MIRA exploiting an N-best representation, and batch lattice MIRA involving hope-and-fear decoding on lattices using oracle lattice decoding algorithm (Chiang et al., 2008). Batch N-best MIRA improves over the MERT weakness when dealing with a large number of feature vectors while still benefiting from the simple alternative decoding-training architecture of MERT. This is the tuning algorithm we use in the remaining of this dissertation.

1.3.2 From tuning algorithms to integrated systems

It is important to notice that the objective functions used in the tuning algorithms described in the previous part can be used, not only to train the log-linear coefficients $\boldsymbol{\lambda}$, but also the parameters of component models which deliver feature vectors $\mathbf{f}(.)$. These tuning procedures could be adapted to become joint procedures that train all free parameters of a SMT system in a unified manner. An advantage of this approach is to avoid the two-phase procedure divided into *training* and *tuning* steps. More remarkably, the ability of incorporating MT quality metrics has become a standard assumption in tuning

algorithms, while the training step generally still uses criteria which have only a loose relationship to the translation performance. An integrated training procedure within a unified framework aims to guide the system parameters to a common objective which is to produce high-quality translations.

The first work in which $\boldsymbol{\lambda}$ are trained on large training data instead of a small development set is (Tillmann and Zhang, 2006). The authors propose a training algorithm for a linearly scored block sequence translation model, which regards the phrase-based SMT system's translation as a sequential process that generates block orientation sequences. The system contains a large set (35M) of easily computed feature functions based on 2-grams block sequences and the orientation of the second block compared to the position of its predecessor. Feature functions here are straightforwardly computed but express poorer information than traditional language and translation models, hence the training of log-linear parameters $\boldsymbol{\lambda}$ is primordial for a good performance. This large-dimension vector $\boldsymbol{\lambda}$ is trained on training data in a global setting to separate block sequences with high BLEU score from block sequences with high model score.

Liang et al. (2006) also train a linear model on large training data with the structured perceptron algorithm. The learning requires to infer a value for latent variables $\mathbf{a}$ for each of the two hypotheses implied in the perceptron update : the "hope", and the "fear" derivations. Another related approach is in (Blunsom et al., 2008) which develops a discriminative version of the hierarchical phrase-based system proposed in (Chiang, 2005), and described in Section 1.1.4. Training consists of maximizing the conditional log-probability $\mathbf{p}(\mathbf{t}|\mathbf{s})$ which is obtained by marginalizing over latent variables' values to compute gradients. Further developments reported in (Blunsom et al., 2008) include a tighter integration with a target language model which is taken into account during the training. On the other hand, the work of (Dyer and Resnik, 2010) focusses on the learning of the reordering model by maximizing the conditional log-probability of the target given the source sentence. The maximum-likelihood objective function is also used in (Lavergne et al., 2011, 2013) where the model takes the form of a Hidden Conditional Random Field (Koo and Collins, 2005; McCallum et al., 2012).

The work in (Yu et al., 2013) is the first time that discriminatively tuning (by an online algorithm) on the training data is shown to significantly outperform conventional methods using only the development set. Instead of choosing between *bold* and *local* updating as in (Liang et al., 2006), the authors propose to use an adaptation of the standard perceptron update in the context of search errors and MT latent variables (the *violation-fixing perceptron* framework (Huang et al., 2012b)). In this approach, oracle translations are computed via forced decoding; the model parameters are updated whenever errors are detected, even though the decoding of the whole sentence has not finished. [10] An advantage of this method, compared to the local updating is that it is theoretically guaranteed to converge and follows more closely the standard perceptron algorithm which encourages updating towards the reference. Moreover the use of a sentence-level BLEU score is no longer needed. The violation-fixing perceptron has recently been applied to the hierarchical phrase-based SMT system (Section 1.1.4) in (Zhao et al., 2014).

[10] This strategy is called *early update* in the paper. Another variant is *max-violation* which consists of waiting for the decoding of the whole sentence finishes, then updating only the worst mistake instead of the first.

Recently, expected BLEU training has been used to train the parameters of a phrase table within the framework of phrase-based SMT. He and Deng (2012) propose a joint discriminative training procedure for phrase and lexicon translation scores. The log-linear coefficients $\boldsymbol{\lambda}$ are still tuned with MERT, alternatively with the expected BLEU training of the translation scores. In (Gao and He, 2013), the same criterion is optimized with SGD, while phrase table scores are modelled based on *Markov Random Field*. Training procedures in the same style for various component models can also be found in (Gao et al., 2014; Auli and Gao, 2014). A common feature of these works is that they propose an *intermediary* approach where $\boldsymbol{\lambda}$ is still tuned separately (although alternatively) using a standard tuning algorithm described in the previous section, only the training step is modified to become a joint training. Although a unified training on large data set using a unique objective function could be more effective, several practical solutions resort to this alternatively organized procedure (which considers the tuning as a black-box process run on held-out development data) because of various reasons :

- Some state-of-the-art tuning algorithms (such as MERT and MIRA) use optimization algorithms that are difficult, even impossible to extend to incorporate other parameters than the log-linear coefficients, mixed together using more complicated schemes than the linear combination;
- Literature on tuning algorithms is abundant [11], whereas the tuning using a standard tool on the same development data would facilitate the comparison with the baseline system;
- Finally, even using a unique training criterion, integrating the optimization of largely different models would still be not straightforward, and would face other problems. For instance, a highly non-linear neural network-based model would be hardly optimized using the same learning rate with the log-linear model within SGD training.

On the other hand, the alternative procedure described in (He and Deng, 2012) requires a consistency between the objective functions used in the discriminative training and tuning. In other words, the discriminative training criterion needs to be elaborated in order to be compatible with the tuning on the development data. This issue will be discussed in more details in Section 5.4.3 of Chapter 5.

1.4 Conclusions

In this chapter, we have presented a brief description of different SMT systems which gives the context for the use of continuous-space models discussed in the rest of the dissertation. Building SMT systems consists of modelling the translation process from training corpora made of set of sentence pairs. These pairs however contain only shallow sentence-level matching between source and target sentences; the learning of hidden relationships inside each sentence pair becomes possible only with the use of latent variables describing the alignment of different components within the source and target sentences, along with

[11] see for example the shared tuning task at WMT'2015 : http://www.statmt.org/wmt15/tuning-task/ .

their translations. These latent variables also help to divide the complex translation process into smaller sub-processes which are easier to model, and which will then be learnt independently from the training data.

In general, existing knowledge about the sub-processes, as well as about different aspects and characteristics of the translation is introduced under the form of feature functions. The building of the whole system is hence based on a two-step procedure. First, several separate models are learnt to give parameters to feature functions; then these functions are combined together within a discriminative *log-linear* framework, in which the features' contributions are called *log-linear coefficients* (Equation (1.2)). The linear combination of feature functions is widely used; however some attempts have been made with a non-linear combination (see for example (Sokolov et al., 2012)).

In practice, different approaches in SMT are characterized by their definition of latent variables, as well as their set of feature functions. Among these approaches, the phrase-based SMT, described mainly in the work of (Zens et al., 2002; Koehn et al., 2003) sketches the standard method of building a translation model from parallel bilingual corpora. Some other approaches, such as the n-gram-based approach (Section 1.1.3) or the hierarchical approach (Section 1.1.4) are variants of the standard technique which try to model the translation differently, or to embody some syntax-based ideas into the phrase-based system.

No matter which approach is used, nearly all SMT systems require to learn the *log-linear coefficients* that combine different feature functions within a log-linear model (Section 1.3.1). This *tuning* step is performed on a *development* set, often separately from the *training* step on training data. In Section 1.3.1, we have seen that all tuning algorithms can be presented as corresponding to different objective functions defined on the development set, although algorithms differ in their optimization method (for instance, gradient descent) and modality (*batch* or *online* learning). In particular, the fore-mentioned criteria can be used, not only to learn the log-linear coefficients, but also to train other parameters from component feature functions, resulting in a globally unified optimization procedure over the entire training data. As the incorporation of MT quality metrics (such as BLEU, Section 1.2) has become a standard assumption in the tuning algorithms, using a joint training procedure may have an advantage over approaches which only uses conventional criteria which have only loose relationship to the translation performance.

The concept of joint training procedures suggests an interesting research direction aiming at improving the training for the component models incorporated in SMT systems. In practice, incorporating and simultaneously training largely different models is not straightforward; practical solutions may resort to an intermediary approach where log-linear coefficients are still tuned separately (although alternatively), only the training step is modified to become a joint procedure (Section 1.3.2). This is the technique we choose for training continuous-space models described in the remaining of this dissertation.

2

Neural Network Language and Translation Models

In the previous chapter, we have provided an overview of current approaches to Statistical Machine Translation (SMT). This chapter is dedicated to a review of different neural network structures used for the language modelling (LM) and translation modelling (TM) tasks. Thanks to these discussions, the chapter helps to clarify what characteristics of continuous-space models (CSMs) in general, and of each particular neural network structure help the model to have a better modelling capacity compared to conventional discrete LMs, and to better handle context words that are important for a good performance in the LM and TM tasks. Besides this, we insist particularly on the difficulty of training and integrating CSMs into the translation process. An overview of the LM task is given in the first section before we proceed to a description of discrete and continuous-space LMs. The application of neural network TMs is presented in Section 2.4, along with a review of different evaluation methods for CSMs.

2.1 The language modelling task

Language model (LM) is an important component of several Natural Language Processing (NLP) applications, such as Automatic Speech Recognition(ASR), or Statistical Machine Translation. The role of this kind of model is to quantify the plausibility of a word sequence, i.e how likely a sequence is in a given language. In some applications, language models are important as they provide a certain prior knowledge about the language we are working with, when other models are less capable of providing or distinguishing a good output from a set of candidates. For example, in an ASR system, while some texts are spelled out similarly and an acoustic model provides little evidence to distinguish between them, such as the two sentences *Let music be the food of love* and *Let music be the foot*

of dove, only a language model trained on a large English corpus can tell us that the first sentence (ending with *love*) is much more likely to be used than the second one.

Given a word sequence w_1^N, a language model assigns a probability $\mathbf{p}(w_1^N)$ to this short piece of text. Its estimation is based on the hypothesis that there exists an unknown distribution from which word strings have been generated. The probability of the sequence w_1^N can be rewritten as :

$$\mathbf{p}(w_1^N) = \prod_{i=1}^{N+1} \mathbf{p}(w_i|w_1^{i-1}) \tag{2.1}$$

which is the product of probabilities computed on each individual word w_i in left-to-right order given all the previous words. In this formulation, we add an additional token </s> at position w_{N+1}) to identify the end of each sentence. By applying the Markov assumption, we can restrict the conditional parts to cover only a limited number of $n-1$ previous words :

$$\mathbf{p}(w_1^N) = \prod_{i=1}^{N+1} \mathbf{p}(w_i|w_{i-n+1}^{i-1}) \tag{2.2}$$

where w_i^j denotes the sequence composed of $w_i, w_{i+1}, ..., w_j$ and w_i is the *begin of sequence* token <s> whenever $i \leq 0$.

This approximation makes the language modelling task much easier compared to the original form of $\mathbf{p}(w_1^N)$ where the random variable w_1^N can have unbounded length. Models using this approximation are called n-gram models; their obvious advantage is the capacity of modelling the original unbounded task with a finite set of parameters. Indeed, suppose that each word w_i takes values from a finite vocabulary $\mathcal{V}$ and the general form $\mathbf{p}(w_i|w_{i-n+1}^{i-1})$ needs to be estimated, we have $|\mathcal{V}|^n$ different values that can be assigned to the sequence w_{i-n+1}^i. Our model can be represented by a set of $|\mathcal{V}|^n$ free parameters, which is huge in practice, but is still a finite set.

The use of a statistical language model is first introduced in the ASR community (Jelinek, 1976) in the context of distinguishing the best outputs from a set of candidates which have similar acoustic scores. Given an acoustic signal $\mathbf{a}$, the application aims at finding the sentence $\mathbf{w}$ that is the most likely to have been spoken. Using the Bayes's theorem, the solution is the one maximizing the following posterior distribution :

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}}\, \mathbf{p}(\mathbf{w}|\mathbf{a}) = \underset{\mathbf{w}}{\operatorname{argmax}}\, \mathbf{p}(\mathbf{a}|\mathbf{w}) \times \mathbf{p}(\mathbf{w}) \tag{2.3}$$

where $\mathbf{p}(\mathbf{a}|\mathbf{w})$ denotes the acoustic model, $\mathbf{p}(\mathbf{w})$ is the LM score corresponding to sentence $\mathbf{w}$. In this framework, the language model reflects a prior knowledge about the language; this information becomes especially relevant for sentences which are undistinguishable by the acoustic model, such as the two sentences "Let music be the food of love" and "Let music be the foot of dove" mentioned above. In SMT, language models play a similar role where the acoustic signal is replaced by a text in a foreign language, and the acoustic model is replaced by a translation model which reflects how likely an English word sequence is a translation of a given source text.

However, the introduction of LMs in these applications needs some careful considerations : whenever the estimated $\mathbf{p}(\mathbf{w})$ is zero, it vanishes all possibility to have $\mathbf{w}$ in the system output no matter of how well it is evaluated by the acoustic or the translation model. A language model which may output zero probabilities runs the risk of making useless the action of other system components. This is the main difficulty when estimating a statistical language model, which becomes extremely severe when training data is always too small to provide reliable estimations to most n-gram sequences. [1] This chapter focusses on the estimation problem, as it represents the major challenge in using language models in practical applications, given that the LM structure will always be the probability of a word given its context composed of either the $n-1$ preceding words (the n-gram model), or of all preceding words (from the beginning of the document, as in some recurrent models).

2.2 Discrete language models

In general, the estimation of a discrete n-gram language model is based on the relative frequencies of word sequences. With an n-gram approximation, all we need to estimate is the probability $\mathbf{p}(w_i|w_{i-n+1}^{i-1})$ of a word w_i given its $n-1$-word context w_{i-n+1}^{i-1}. The maximum likelihood estimation (MLE) results in a relative frequency-based method to estimate this distribution. Let $c(w_i^j)$ be the number of occurrences of the sequence w_i^j in a training corpus, then the MLE gives the following estimate :

$$\mathbf{p}_{MLE}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})}$$

This estimation is however often not statistically reliable for all contexts, especially for applications in which there is a great number of different contexts. It is not guaranteed to observe each such sequence, even once, to derive a reliable estimation. A historical example is considered in (Rosenfeld, 1996), where, after observing all trigrams ($n=3$) in a text of 38 million words from the Wall Street Journal corpus, the author realizes that a third of all trigrams on held-out news articles are still unseen. The problem becomes more severe with rare words and sequences, especially when it is desirable to increase n to handle a larger contextual information. Indeed, the problem lies in the number of model free parameters which increases exponentially with n. At a consequence, the MLE needs to be adjusted with some smoothing techniques that will be described in the next section.

2.2.1 Smoothing techniques

Smoothing techniques aim at adjusting the described MLE probabilities in order to redistribute the probability mass to rare word sequences. Not only do they suppress zero probabilities, but they also improve the general quality of the probability estimation by introducing a generalisation mechanism.

[1] which is easily the case with high-order models.

As mentioned in (Kneser and Ney, 1995), all smoothing techniques via **backoff** can be described by the following equation :

$$\mathbf{p}_{smooth}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \alpha(w_i|w_{i-n+1}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \gamma(w_{i-n+1}^{i-1})\beta(w_i|\{w_{i-n+1}^{i-1}\}) & \text{if } c(w_{i-n+1}^{i}) = 0 \end{cases} \tag{2.4}$$

It means we divide the estimation of $\mathbf{p}(w_i|w_{i-n+1}^{i-1})$ in two cases depending on the absolute count of the context sequence in training data. In the case of non-zero count, the probability $\alpha(w_i|w_{i-n+1}^{i-1})$ is established mainly from the relative count $c(w_{i-n+1}^{i})/c(w_{i-n+1}^{i-1})$, which is then discounted by a discount ratio (Good-Turing estimate (Good, 1953), Katz Smoothing (Katz, 1987)), or by a fixed discount $D < 1$ (Absolute Discounting (Ney et al., 1994), Kneser-Ney Smoothing (Kneser and Ney, 1995)). The discounted probability mass is then redistributed among those with zero counts through a backoff step. This backing-off resorts to a distribution β which conditions on an equivalence class $\{w_{i-n+1}^{i-1}\}$, less specific than the context itself, in order to have more reliable estimate of unseen events. Typically, the model considers a lower order context w_{i-n+2}^{i-1}. The normalization coefficient $\gamma(w_{i-n+1}^{i-1})$ is present to ensure all terms in (2.4) sum to one.

Another approach is to linearly **interpolate** higher- and lower-order n-gram models (Bell et al., 1990; Witten and Bell, 1991), so that in all cases, the smoothed probability $\mathbf{p}_{smooth}(w_i|w_{i-n+1}^{i-1})$ involves higher-order count and lower-order estimates :

$$\mathbf{p}_{smooth}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})} + (1 - \lambda_{w_{i-n+1}^{i-1}})\mathbf{p}_{smooth}(w_i|w_{i-n+2}^{i-1})$$

The key difference between *backoff* and *interpolated* models hence lies in the estimation of non-zero-count events. Indeed, while interpolated models always use information from lower-order distributions, backoff models do not. In (Chen and Goodman, 1996), the authors remark that it is easy to create a backoff version of an interpolated model, and the interpolated and backoff versions of some existing algorithms have been implemented and tested. In that work, they also propose a methodology aiming at fairly comparing the performance of different smoothing techniques proposed in (Jelinek, 1980; Katz, 1987; Bell et al., 1990; Ney et al., 1994) and (Kneser and Ney, 1995). A modified version of Moreover, the authors propose a modified version of the *Kneser-Ney* method (Kneser and Ney, 1995), which outperforms all other methods, has also been proposed. The procedure, commonly known in literature as the *modified Kneser-Ney smoothing*, is a recommended choice for many applications that imply the use of a language model estimation technique.

2.2.2 Class-based models

Another way to overcome the difficulty due to large vocabularies and rare words when estimating an n-gram language model is to use word clustering. This approach (Ward et al., 1990; Brown et al., 1992; Kneser and Ney, 1993) relies on the idea that information learnt from the occurrences of a given word can be shared among other words that belong to the same group. In the general form, the first step consists of assigning a class C_i to each word w_i in the vocabulary. Then, the n-gram probability $\mathbf{p}(w_i|w_{i-n+1}^{i-1})$ can be

estimated in some different ways (Rosenfeld, 2000), for instance with a trigram model :

$$\mathbf{p}(w_i|w_{i-1}, w_{i-2}) = \mathbf{p}(w_i|C_i)\mathbf{p}(C_i|w_{i-1}, w_{i-2}) \tag{2.5}$$
$$\mathbf{p}(w_i|w_{i-1}, w_{i-2}) = \mathbf{p}(w_i|C_i)\mathbf{p}(C_i|w_{i-1}, C_{i-2}) \tag{2.6}$$
$$\mathbf{p}(w_i|w_{i-1}, w_{i-2}) = \mathbf{p}(w_i|C_i)\mathbf{p}(C_i|C_{i-1}, C_{i-2}) \tag{2.7}$$
$$\mathbf{p}(w_i|w_{i-1}, w_{i-2}) = \mathbf{p}(w_i|C_{i-1}, C_{i-2}) \tag{2.8}$$

Class-based models represent an attempt to exploit the relationships between vocabulary words throughout the use of clusters, instead of considering them solely as elements in a discrete finite vocabulary set. Such relationships can also be expressed in terms of morphological, syntactic or semantic properties. For instance, the occurrence of a trigram such as *I like cat* will certainly give some insight regarding the likelihood of another trigram *I like dog*, as the two final words *dog* and *cat* play a similar role in this context. Intuitively, when a language model gives a non-zero probability to the occurrence of *cat* just after *I* and *like*, it should not give zero probability to $\mathbf{p}(dog|like, I)$, even if the trigram is not observed. Class-based models help to formalize such intuition by first clustering *cat* and *dog* in the same cluster, then whenever an n-gram occurs which contains either word from this cluster, it will transfer information and redistribute the probability to other words in the cluster.

An obvious weakness of this kind of models may be the fact that they depend upon the clustering of words. This structure can be learnt carefully from the training data (Brown et al., 1992; Kneser and Ney, 1993), or be decided based on some external linguistic knowledge (Ward et al., 1990). Another weakness is that they are often based on *hard* clustering which expresses the word-to-word relationship in a strict manner under the assumption that each word belongs to one and only one cluster. In practice, words can be related to each other via different aspects (semantics, syntax or morphology); words can also be grouped into multiples categories. Continuous-space language models described in Section 2.3 constitutes a promising alternative that gives a more flexible mean to exploit these relationships between words or linguistic units through the concept of word continuous representation. At the core of the approach, semantic and syntactic relatedness is to be learnt implicitly through the projection of all vocabulary words into a multi-dimensional continuous space in which inter-word relationships are expressed in terms of distances between word embeddings.

2.2.3 Linguistically motivated models

All the techniques presented in the previous section have in common that they use little knowledge of what the language really is. Indeed, these methods could be applied as well to sequences of arbitrary symbols as to words or characters. This lack of linguistic knowledge is an obvious direction for further improvements, and incorporating strong linguistic knowledge seems to be a promising approach. One of the first attempt along these lines is the use of language models that are directly derived from grammars, such as Context free (CFG) or Link grammar. An overview of such models can be found in (Rosenfeld, 2000).

Another approach that still falls into the category of linguistically motivated models,

but that can be estimated quite efficiently using n-gram techniques is the Dependency (or Structured) language models of (Chelba et al., 1997; Chelba and Jelinek, 2000). Dependency grammars (DG) describe sentences in terms of asymmetric pairwise relationships among words. Each word in the sentence is considered to be dependent upon one other word that is called *head* or *parent*; the single exception is the *root* which serves as the head of the entire sentence. A probabilistic version of DG has also been developed (Carroll and Charniak, 1992), which provides a suitable tool for the language modelling task using usual techniques of n-gram LMs. Instead of conditioning on a few preceding words ($n-1$-gram context), probabilistic DG has an additional latent variable representing a dependency graph which decides which words are to be taken into the context for the word prediction. The latent variable, throughout the concept of dependency graph, chooses among all words that have occurred since the beginning of the sentence only the most relevant elements that can help to boost the prediction capacity. Irrelevant context words are ignored, the model hence uses more efficiently context information while still limiting itself to an n-gram model. Examples of this idea can be found in (Chelba et al., 1997; Chelba and Jelinek, 2000), or more recently in (Gubbins and Vlachos, 2013).

2.2.4 Overview on discrete language models

There are other ideas on discrete language models that have not been mentioned here, such as the adaptive models (or Global semantics language models) which are based on the assumption that documents may differ in domains, topics and styles (Gildea and Hofmann, 1999; Bellegarda, 2000; Wang et al., 2003; Tam and Schultz, 2005, 2006; Watanabe et al., 2011), or the exponential models which allow us to incorporate arbitrary feature functions representing knowledge from different sources, including linguistic knowledge (Lau et al., 1993; Berger et al., 1996; Chen and Rosenfeld, 2000).

However, a common feature of all the LMs described so far is their being based on statistics of discrete symbols (from a finite set which is the vocabulary) which forces the estimation techniques for these models to deal with severe problems related to *data sparsity*. As a consequence, smoothing techniques are employed to correct the MLE probabilities derived from relative counts. These methods are often based on various tricks to redistribute the probability mass to rare events which are undermined by the direct count. In terms of modelling task, the conventional n-gram discrete language model contains a number of free parameters which increases exponentially with n. Reducing this quantity might be the key to improve LMs. This reduction can be realized throughout the principle of *sharing parameters* : word sequences are modelled using the information from the words they are composed of; then whenever a new sequence needs to be handled, the model can reuse parameters from other sequences, instead of considering it as a totally new event. Class-based LMs provide an illustration of this idea in spite of the fact that their probabilities still rely on relative counts (of word classes). Continuous-space LMs, as described in the next section, give an example in which the representation of a sequence may be as simple as the concatenation of representations from its sub-parts. Not only the number of parameters is reduced, the models also produce by default only non-zero probabilities, even though a MLE estimate is still employed. [2]

[2] with a simple regularization term whose the role is not as significant as in smoothing techniques for

2.3 Continuous space language models

In this section, we present an overview of continuous-space models (CSMs) for the language modelling task. The term *continuous-space models* often refers to those exploiting continuous representation instead of discrete-element-based representation as do the models presented in the previous sections. Continuous representation can be expressed within different frameworks. For instance, Bellegarda (1997) use successfully Latent Semantic Analysis to derive a continuous representation of words. Sarikaya et al. (2008) propose to obtain these word embeddings via an adaptation of Latent Semantic Analysis, where a Gaussian mixture model is learnt on the upper layer within a hidden Markov model.

However, in the scope of this dissertation, we focus on the description of continuous-space models based on artificial neural networks. An advantage of this category of models is that word representations are expressed in terms of weights inside a neural network, and are learnt through the training process of the neural network. The projections and all inter-word relationships are hence learnt implicitly and jointly with all other parameters on the training corpora, rather than being inherited from an external model. This feature makes neural network-based language models (NNLMs) useful in two aspects. On the one hand, it can be incorporated to applications requiring a LM (such as ASR, SMT, etc.). On the other hand, it can also be seen as being a preliminary phase to train continuous word representations which are then used in other applications, as typically proposed in (Collobert and Weston, 2008; Turian et al., 2010).

This section begins with a description of some neural network structures used for the language modelling task, starting with the conventional feed-forward model proposed in (Bengio et al., 2003a), then proceeding to other structures such as recurrent neural networks. An important part of this report is dedicated to the description of some techniques that aim to overcome the computational difficulty related to the training and inference, especially at the output layer.

2.3.1 Conventional feed-forward model

The work of (Bengio et al., 2003a) is not the first time neural networks (NNs) have been applied to the language modelling task. The idea has been inspired from previous works in which distributed representation for discrete symbols have been used within a neural network (Hinton, 1986; Elman, 1990; Paccanaro and Hinton, 2000). The use of NNs for the language modelling task has also been investigated in (Nakamura et al., 1990; Miikkulainen and Dyer, 1991; Schmidhuber and Heil, 1996; Xu and Rudnicky, 2000). But (Bengio et al., 2003a) has been the first work in which the idea has been applied to large-scale data. A peculiarity of this model is the joint training of two sets of parameters : the first set contains distributed *word feature vectors*, while the second set represents a function which outputs estimates for n-gram sequence probabilities. The goal is to model n-gram probabilities $\mathbf{p}(w_i|w_{i-n+1}^{i-1})$, where words w_i can take values from a finite vocabulary $\mathcal{V}$. The first set of parameters expresses a mapping function $\mathbf{C}$ from any element $w \in \mathcal{V}$ to a multi-dimensional vector $\mathbf{C}(w) \in \mathbb{R}^{\mathrm{D}}$. In practice, $\mathbf{C}$ is represented by a $\mathrm{D} \times |\mathcal{V}|$ matrix,

discrete LMs.

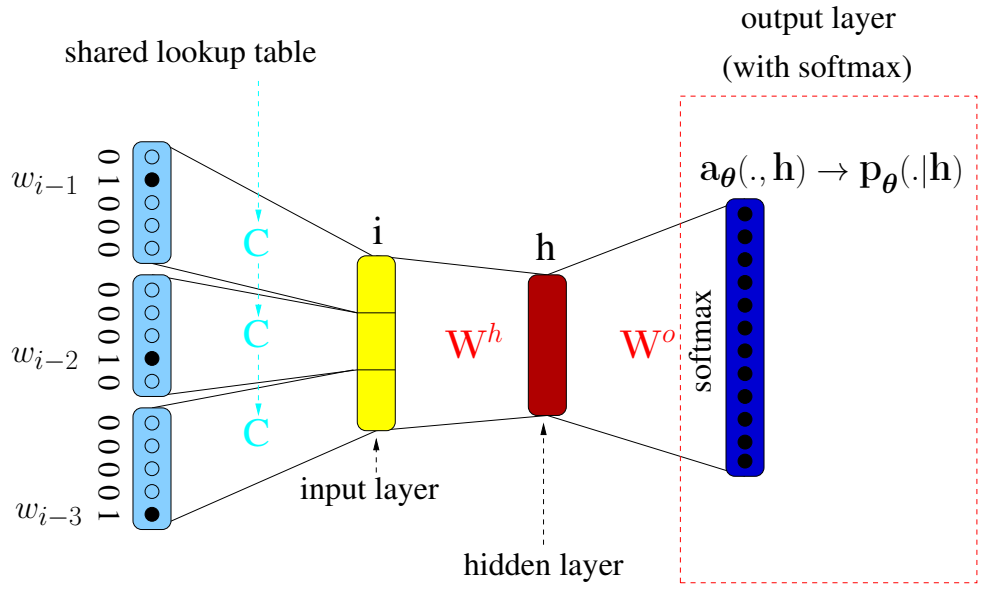


Figure 2.1 – The feed-forward neural network language model presented in (Bengio et al., 2003a). **h** denotes the hidden layer's activation which provides a compact representation of context words, hence the probability $\mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c})$ can be rewritten as $\mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{h})$.

each column represents the real multi-dimensional vector corresponding to a word in $\mathcal{V}$. In order to obtain the embedding of w, one needs to find the column corresponding to w inside this matrix, also called the *lookup table*. The operation can also be viewed as the product of $\mathbf{C} \times \mathbf{l}_w$, where $\mathbf{l}_w$, called a *one-hot vector*, is a $|\mathcal{V}|$-dimensional vector in which all components are zero except the one corresponding to w.

The second set of parameters is a function which, given the word embeddings of w_{i-n+1}^{i-1} and the predicted word w_i, outputs the estimated probability of the word w_i given its context w_{i-n+1}^{i-1}. Let $\boldsymbol{\theta}$ be the set of all free parameters (which is the union of the two afore-mentioned sets), the estimated probability can be written as :

$$\mathbf{p}_{\boldsymbol{\theta}}(w_i|\mathbf{C}(\mathbf{c}_{ngram}(w_i))) \text{ or } \mathbf{p}_{\boldsymbol{\theta}}(w_i|\mathbf{c}_{ngram}(w_i))$$

where $\mathbf{c}_{ngram}(w_i) := w_{i-n+1}^{i-1}$ represents the context words of w_i within the n-gram framework, and $\mathbf{C}(\mathbf{c}_{ngram}(w_i))$ is the concatenation of the word embeddings of $w_{i-n+1}, ..., w_{i-1}$.

We can view $\mathbf{C}(\mathbf{c}_{ngram}(w_i))$ as one single feature vector that contains information about all context words. The input is simply the concatenation of all context embeddings, hence giving a *flat* representation about the context. The vector is then applied to a sequence of neural network layers to deliver more and more compact representations of the context. The function $\mathbf{p}_{\boldsymbol{\theta}}(.)$ is modelled as a neural network containing an input, a hidden and an output layer, as represented in Figure 2.1.

The input layer

The vector $\mathbf{C}(\mathbf{c}_{ngram}(w_i))$ is fed to a standard feed-forward neural network via an input layer **i** of $(n-1)$D units. To obtain this vector, an additional neural network layer is added at the beginning of the chain, taking as input the concatenation of $n-1$ one-hot

vectors of the context words :

$$\{\mathbf{l}_{w_{i-n+1}}, \mathbf{l}_{w_{i-n+2}}, ..., \mathbf{l}_{w_{i-2}}, \mathbf{l}_{w_{i-1}}\} \tag{2.9}$$

and as weight matrix $n-1$ replicates of the matrix $\mathbf{C}$, which produces the vector $\mathbf{C}(\mathbf{c}_{ngram}(w_i)) = \{\mathbf{C} \times \mathbf{l}_{w_{i-n+1}}, ..., \mathbf{C} \times \mathbf{l}_{w_{i-1}}\}$ as output. This is a conventional neural network layer, except that all parameters in $\mathbf{C}$ are shared among $(n-1)$ parts of the weight matrix.

The hidden layer

After obtaining the input $\mathbf{i} = \mathbf{C}(\mathbf{c}_{ngram}(w_i))$, the hidden layer, defined by a weight matrix $\mathbf{W}^h \in \mathbb{R}^{\mathrm{H} \times (n-1)\mathrm{D}}$ and a bias vector $\mathbf{b}^h \in \mathbb{R}^{\mathrm{H}}$ transforms $\mathbf{i}$ into an output activation :

$$\mathbf{h} = \mathrm{f}\left(\mathbf{W}^h \times \mathbf{i} + \mathbf{b}^h\right) \tag{2.10}$$

where f is a (non-linear) function, which can be tangent hyperbolic ($\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$), or sigmoid function ($\mathrm{sigm}(x) = \frac{1}{1+\exp(-x)}$). The vector $\mathbf{h} \in \mathbb{R}^{\mathrm{H}}$ can be considered as a more abstract and more compact representation of the context words.

The output layer

From the context representation $\mathbf{h}$, the output layer computes an output vector $\mathbf{a} = \mathbf{a}_{\boldsymbol{\theta}}(., w_{i-n+1}^{i-1}) \in \mathbb{R}^{|\mathcal{V}^o|}$, each component of which corresponds to a word $w \in \mathcal{V}^o$ evaluating the likeliness of the word w_i in the context of $w_{i-n+1}, ..., w_{i-1}$:

$$\mathbf{a} = \mathbf{W}^o \times \mathbf{h} + \mathbf{b}^o \tag{2.11}$$

where $\mathbf{W}^o \in \mathbb{R}^{|\mathcal{V}^o| \times \mathrm{H}}$ and $\mathbf{b}^o \in \mathbb{R}^{|\mathcal{V}^o|}$ are the free parameters of the layer. Here, the output vocabulary $\mathcal{V}^o$ can be different from the context vocabulary $\mathcal{V}$. A such generalization is necessary when we consider a *short-list* approach (Section 2.3.4), or when the conventional NNLM is extended to a bilingual continuous-space translation model (Section 2.4).

In order to have a probabilistic interpretation on $\mathcal{V}^o$, we take the exponentials of activations, then normalize them through a *softmax* function :

$$\mathbf{p}_{\boldsymbol{\theta}}(w_i | w_{i-n+1}^{i-1}) = \frac{\exp(\mathbf{a}_{\boldsymbol{\theta}}(w_i, w_{i-n+1}^{i-1}))}{\sum_{w \in \mathcal{V}^o} \exp(\mathbf{a}_{\boldsymbol{\theta}}(w, w_{i-n+1}^{i-1}))} \tag{2.12}$$

where $\mathbf{a}_{\boldsymbol{\theta}}(w, w_{i-n+1}^{i-1})$ denotes the component of vector $\mathbf{a}$ corresponding to the word $w \in \mathcal{V}^o$.

Table 2.1 summarizes the notations used in this section; some of them will also be used later to describe other neural network structures. The standard model has three hyper-parameters : the dimension of the word feature vectors D, the hidden layer size H and the non-linear function f . The two sets of parameters described above are gathered in a vector $\boldsymbol{\theta}$ including : the mapping matrix (or lookup-table) $\mathbf{C}$, the weight matrix $\mathbf{W}^h$ and bias vector $\mathbf{b}^h$ of the hidden layer, and the weight matrix $\mathbf{W}^o$ and bias vector $\mathbf{b}^o$ of

Notation	Meaning
n	Degree of the (n-gram) model
$\boldsymbol{\theta}$	Vector containing all free parameters of the network
D	Dimension of word embeddings
$\mathbf{c}_{ngram}(w)$, $\mathbf{c}_{recc}(w)$	Context of a word w in the framework of an n-gram ($ngram$) or a recurrent ($recc$) model
$\mathbf{l}_w$	One-hot vector corresponding to a word w
$\mathbf{C}$, $\mathbf{C}(w)$	Lookup-table matrix, feature vector (embedding) of w
$\mathbf{i}$	$(n-1)$D-dimensional activation of the input layer
$\mathbf{W}^h$, $\mathbf{b}^h$	Weight matrix and bias vector of the hidden layer
f	Non-linearity function
$\mathbf{h}$	Activation of the hidden layer, which is the most compact representation of the context
H	Dimension of $\mathbf{h}$
$\mathbf{W}^o$, $\mathbf{b}^o$	Weight matrix and bias vector of the output layer
$\mathcal{V}$, $\mathcal{V}^o$, $\lvert\mathcal{V}^o\rvert$	Input and output vocabularies, the size of the output vocabulary
$\mathbf{a}$, $\mathbf{a}_{\boldsymbol{\theta}}(w, \mathbf{c})$	Activation vector of the output layer and its component corresponding to w, these variables are computed from $\mathbf{c}$
$\mathbf{e}$, $\mathbf{e}_{\boldsymbol{\theta}}(w, \mathbf{c})$	Exponential of the output layer activation, $\mathbf{e} = \exp(\mathbf{a})$
$\mathbf{p}_{\boldsymbol{\theta}}(w\lvert\mathbf{c})$	Probability of the word w appearing after a context $\mathbf{c}$, estimated by the neural network
$g_{\boldsymbol{\theta}}(w, \mathbf{c})$	Output score of the neural network for the n-gram $(\mathbf{c}, w)$, which is the log-probability of the last word given its context
$\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c})$	Partition function, computed for each context $\mathbf{c}$, and depends on $\boldsymbol{\theta}$

Table 2.1 – Notations used for describing standard NNLMs.

the output layer. In practice, when evaluating an NNLM with *perplexity*, or using it in an application such as SMT (see Section 2.5), we often use the log-probability. It is hence convenient to explicitly define the output score as :

$$\begin{aligned} g_{\boldsymbol{\theta}}(w_i, w_{i-n+1}^{i-1}) &= \log \mathbf{p}_{\boldsymbol{\theta}}(w_i | w_{i-n+1}^{i-1}) \\ &= \mathbf{a}_{\boldsymbol{\theta}}(w_i, w_{i-n+1}^{i-1}) - \log \left(\sum_{w \in \mathcal{V}^o} \exp(\mathbf{a}_{\boldsymbol{\theta}}(w, w_{i-n+1}^{i-1})) \right) \\ &= \mathbf{a}_{\boldsymbol{\theta}}(w_i, w_{i-n+1}^{i-1}) - \log(\mathrm{H}_{\boldsymbol{\theta}}(w_{i-n+1}^{i-1})) \end{aligned} \tag{2.13}$$

where $\mathrm{H}_{\boldsymbol{\theta}}(.)$ denotes the normalization constant which is the sum of $\exp(\mathbf{a}_{\boldsymbol{\theta}}(w, .))$ over all $w \in \mathcal{V}^o$, and which depends on the network input (i.e the context). Here we add the subscript $\boldsymbol{\theta}$ to indicate the dependence on free parameters.

In this model, the two matrices $\mathbf{C}$ and $\mathbf{W}^o$ play similar roles as they define maps between vocabularies ($\mathcal{V}$ and $\mathcal{V}^o$) and continuous spaces. Hence we can refer to $\mathbf{C}$ as the lookup table projecting to the *input (context) space*, and to $\mathbf{W}^o$ as projecting to the *output (prediction) space*. The two matrices can share the same parameter values, in conditions that the two vocabularies are the same, $\mathcal{V} = \mathcal{V}^o$ and $\mathrm{D} = \mathrm{H}$. In this situation, we obtain the *log-bilinear model* proposed in (Mnih and Hinton, 2007). This model further reduces the number of trainable parameters, compared to the standard version.

2.3.2 Recurrent neural network language model

A major drawback of the conventional model described in (Bengio et al., 2001) is that the feed-forward structure imposes a fixed-length context, which limits the amount of contextual information exploited by the language model. For discrete language models, because of the data sparsity problem and in order to have reliable statistics, the number of context words is often limited to 4. Attempts have been made to handle longer dependencies, such as the Structured language models (Chelba et al., 1997; Chelba and Jelinek, 2000). For CSMs, Mikolov et al. (2010) show that it is possible to implement a structure called *Recurrent neural network* in order to construct a compact representation for all context words from the beginning of a document until the current word to be predicted.

The basic implementation in (Mikolov et al., 2010) is based on an Elman network (Elman, 1990). Here, the key modification lies in the hidden activation $\mathbf{h}$, which should contain information of all words since the beginning of the document (or from the token <s> of each sentence) until the last word preceding the current predicted word. The most straightforward method is to compute $\mathbf{h}$ recursively for the current position i as follows :

$$\mathbf{h}_i = \mathrm{f}\left(\mathbf{W}^h \times \mathbf{h}_{i-1} + \mathbf{C} \times \mathbf{l}_{w_{i-1}}\right) \tag{2.14}$$

where, $\mathbf{h}_{i-1}$ is the hidden activation vector used for predicting the word w^{i-1} [3], and $\mathbf{W}^h$ is still the weight matrix of the hidden layer which is time-independent (see Table 2.1). The term $\mathbf{C} \times \mathbf{l}_{w_{i-1}}$, as described in the previous section, is the feature vector of w_{i-1}. As for the standard structure, the probability $\mathbf{p}_{\boldsymbol{\theta}}(w_i | \mathbf{c}_{recc}(w_i))$ is still computed from the

[3] which means that it captures information about context words until w_{i-2}.

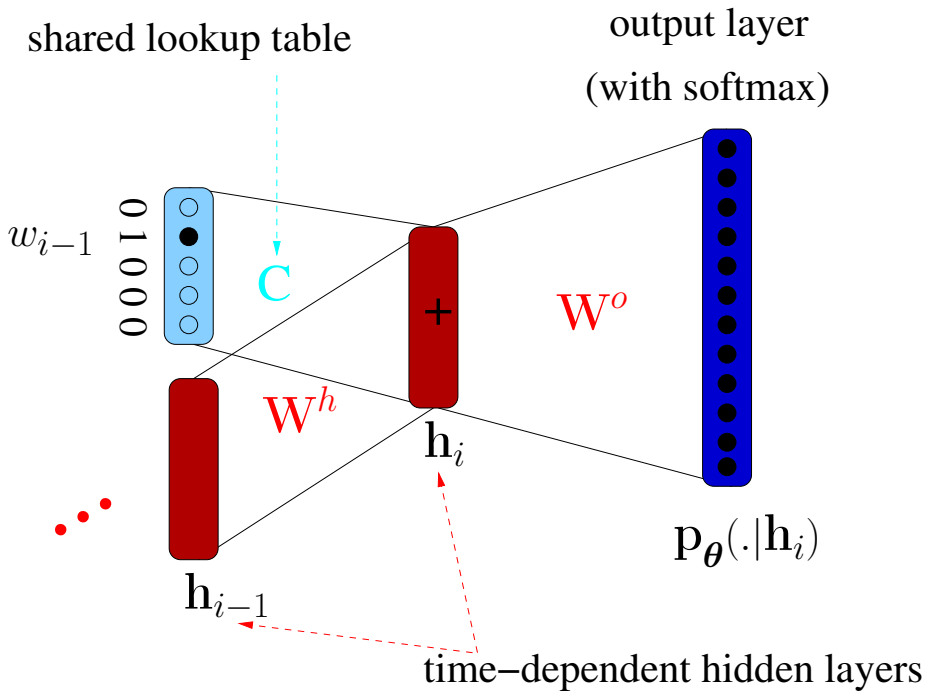


Figure 2.2 – The compact structure of a recurrent NNLM.

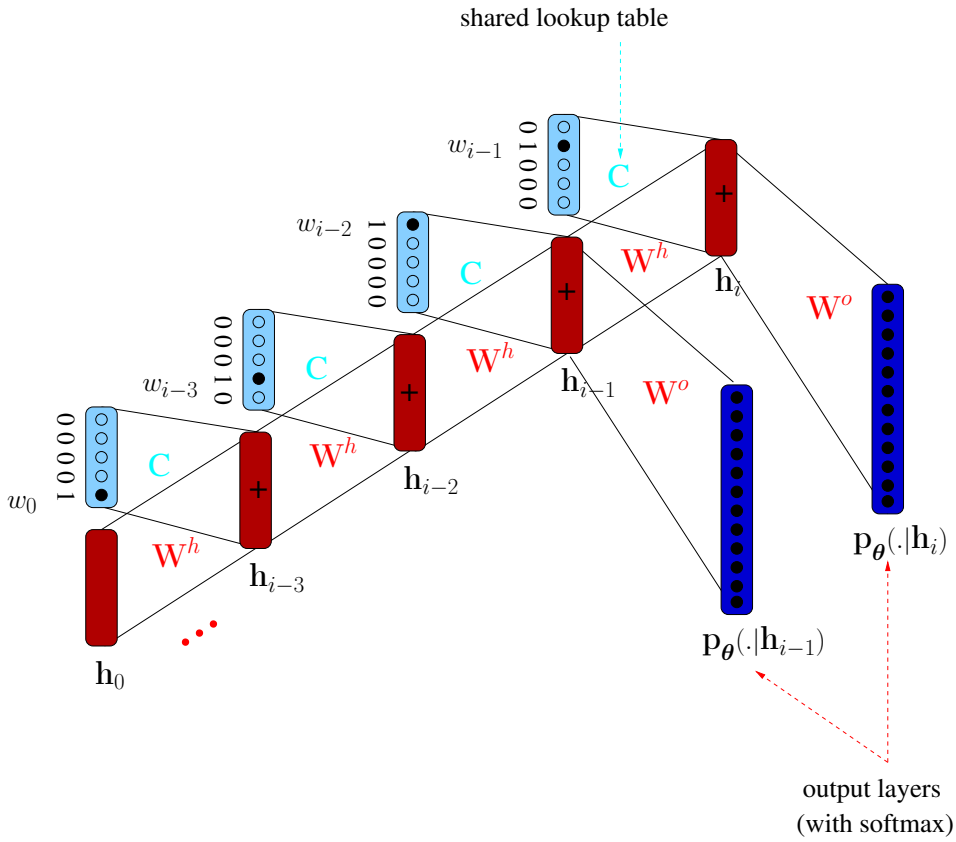


Figure 2.3 – The unrolled recurrent NNLM.

hidden activation vectors using (2.11) and (2.12) :

$$\mathbf{a}^i = \mathbf{W}^o \times \mathbf{h}_i + \mathbf{b}^o$$
$$\mathbf{p}_{\boldsymbol{\theta}}(w_i|\mathbf{c}_{recc}(w_i)) = \frac{\exp(\mathbf{a}^i_{\boldsymbol{\theta}}(w_i, \mathbf{c}_{recc}(w_i)))}{\sum_{w \in \mathcal{V}^o} \exp(\mathbf{a}^i_{\boldsymbol{\theta}}(w, \mathbf{c}_{recc}(w_i)))}$$

where we add the subscript i to indicate the dependency on time i (all quantities without subscript i are time-independent), and $\mathbf{c}_{recc}(w_i)$ is the context of a recurrent neural network. The whole model can be described compactly as shown in Figure 2.2. The training algorithm is *truncated back-propagation* in which the lookup table matrix $\mathbf{C}$, the weight matrix and bias vector of the hidden layer $\mathbf{W}^h$ and $\mathbf{b}^h$ are updated at time i using the error gradient computed for the current time step. The algorithm can be viewed just as *back-propagation* applied on the unrolled model of Figure 2.2.

Compared to the basic version in (Mikolov et al., 2010), Mikolov et al. (2011b) propose several modifications among which the most relevant is the use of (truncated) Back-Propagation Through Time (BPTT) (Rumelhart et al., 1985). BPTT consists of unrolling the network (Figure 2.3) from its compact form (Figure 2.2) and back-propagating error gradients through multiple time steps. The unrolled version shows the impact between the first word <s> and the current prediction; however parameter updates are truncated upon a threshold τ. Hence, the truncated back-propagation in the first work can be considered as BPTT with $\tau = 1$ as if words laying far away backward no longer have an impact on the current word w_i. Additional information and a practical guide on truncated BPTT can be found in (Boden, 2001). A comparison between recurrent and feed-forward neural network NNLMs has also been presented in (Mikolov et al., 2011b). According to their experiments, the perplexity obtained by a mixture of recurrent NNLMs is shown to be better than the one with feed-forward NNLMs. Other comparative results in ASR applications are reported in (Mikolov et al., 2011a; Sundermeyer et al., 2013).

Context-dependent recurrent NNLM

We have seen so far that the hidden activation vector $\mathbf{h}_i$ (also referred to as *hidden state*) in the recurrent NNLM is a time-dependent quantity which accumulates the information of an additional context word after each time step. This vector can however incorporate other information than the presence of words, by extending the hidden layer input to also include an *auxiliary input*, which transforms Equation (2.14) to the new following form :

$$\mathbf{h}_i = \mathbf{W}^h \times \mathbf{h}_{i-1} + \mathbf{C} \times \mathbf{l}_{w_{i-1}} + \mathbf{F} \times \mathbf{f}_i \tag{2.15}$$

The corresponding structure is displayed in Figure 2.4.

The time-dependent auxiliary input $\mathbf{f}_i$ may in theory represent any context information other than the recurrent context $\mathbf{c}_{recc}(w_i)$. The first attempt to exploit this additional piece of context is in (Mikolov and Zweig, 2012) where the authors propose to feed into $\mathbf{f}_i$ a global context feature to further reinforce the ability of recurrent NNLMs to capture longer dependencies. This feature vector, then conventionally interpreted as a topic representation, is concatenated to the last word feature $\mathbf{C} \times \mathbf{l}_{w_{i-1}}$ and the previous hidden vector $\mathbf{h}_{i-1}$ as described in Figure 2.4; the resulting hidden state is used to predict

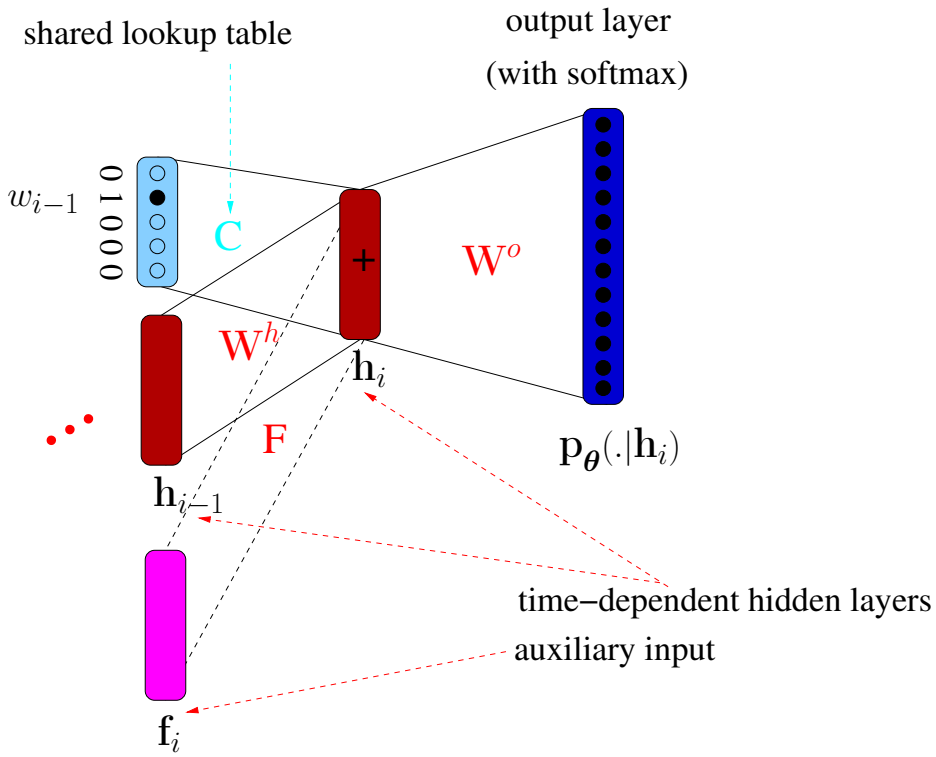


Figure 2.4 – The compact structure of a context-dependent recurrent NNLM, which involves an *auxiliary input layer* $\mathbf{f}_i$.

the current word w_i. The topic representation vector is estimated, at each position i, from a fixed-length word block preceding the current word, using Latent Dirichlet Allocation (Blei et al., 2003). The proposal can be described as a combination of recurrent NNLMs with the adaptive models (Gildea and Hofmann, 1999; Bellegarda, 2000; Wang et al., 2003; Tam and Schultz, 2005, 2006; Watanabe et al., 2011), except that the added topic feature vectors here are computed from a fixed-length window (50 preceding words in the experiments). The generalization capacity of continuous-space models also allows the authors to train a joint model which avoids having to partition the training data into subsets of different topics (on which multiple models have to be built) which often leads to data fragmentation. The proposal has been reported to result in the lowest published perplexity on the Penn Treebank data, and to WER improvements for the Wall Street Journal ASR task. The auxiliary input vector later plays a crucial role for building recurrent bilingual CSTMs from recurrent NNLMs (Section 2.4.3).

Vanishing gradients and LSTM networks

The gains obtained by adding (global) context feature vectors described above give a quite confusing message, as the recurrent NNLM is assumed to have knowledge about all context words, since the beginning of the document until the last word preceding the current position. Experimental results show that, in spite of the recurrent design, this kind of structure always suffers from estimation problems due to long-term dependencies, for the gradient computation becomes increasingly ill-behaved when errors must be propagated further back in time. This problem is often referred to as the problem of *vanishing*

gradient (Bengio et al., 1994; Hochreiter and Schmidhuber, 1997).

Practical difficulties when using gradient descent, either by truncated back-propagation or by BPTT to optimally train recurrent neural networks have been mentioned in early work, for instance in (Bengio et al., 1994). Experimental evidences suggest that training with gradient descent often results in sub-optimal solutions which take into account only short-term dependencies, but ignore long-term dependencies (Bengio et al., 1992; Mozer, 1993). The main reason is that when using the tangent hyperbolic or the sigmoid as the non-linear function f , the derivative of the objective function $\mathcal{L}^i(\boldsymbol{\theta})$ (computed when predicting w_i) with respect to the hidden state vector $\mathbf{h}_{i-k}$ (at k steps backward) is scaled at each backward step, hence either exponentially converges or explodes as k increases. These gradients (indexed by i and k) in turn identify the update direction for all network free parameters, such as the shared weight matrix $\mathbf{W}^h$, for which gradient is computed by :

$$\frac{\partial \mathcal{L}^i(\boldsymbol{\theta})}{\partial \mathbf{W}^h} = \sum_{k=0}^{i-1} \frac{\partial \mathcal{L}^i(\boldsymbol{\theta})}{\partial \mathbf{h}_{i-k}} \times \mathbf{h}_{i-k-1}^T$$

An observation can be made from the above formula is that $\mathbf{W}^h$ is updated towards a direction along which long-term effects (i.e with $k \gg 1$) either vanish or explode; both situations lead to a scaling problem in updating $\mathbf{W}^h$. It simply means that $\mathbf{W}^h$ is much more easily optimized to reflect short-term than to reflect long-term dependencies between output predictions and hidden states, and that if the training converges, the final model may not express richer modelling information than the simpler n-gram structure. The same holds for the update of other model parameters.

In order to improve the capacity of recurrent NNLMs in remembering long-term effects, besides the explicit concatenation of global context information to the network hidden state as described above, other solutions can be divided in two groups, either by using alternative algorithms to the gradient descent, or by modifying the non-linear function f (which often results in a modification of the network structure) in order to prevent the gradient from vanishing. In the first group, some alternative solutions to plain SGD have been proposed, including Simulated Annealing and Discrete Error Propagation (Bengio et al., 1994), explicitly introduced time delays (Lang et al., 1990; Lin et al., 1996) or time constants (Mozer, 1993), hierarchical sequence compression (Schmidhuber, 1992), and Hessian-free optimization (Martens, 2010; Martens and Sutskever, 2011; Sutskever et al., 2011). However, they are either particularly expensive because of the use of higher-order gradient information, or the requirement of a batch optimization algorithm is not suited to practical systems trained on huge training corpora. With the view of applying RNNs to the language modelling task, the second group, which consists of modifying the network hidden layer while still keeping unchanged the training algorithm, seems to be a more promising approach. Several implementations along this research line have been experimented, including the LSTM (Hochreiter and Schmidhuber, 1997) or a simpler version recently proposed in (Cho et al., 2014).

Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997; Gers et al., 2003) is a recurrent architecture specifically designed to address the vanishing gradient problem, while still using BPTT for training. The idea of LSTM is to re-design the structure in such a way that the scaling factor at each backward propagation is fixed

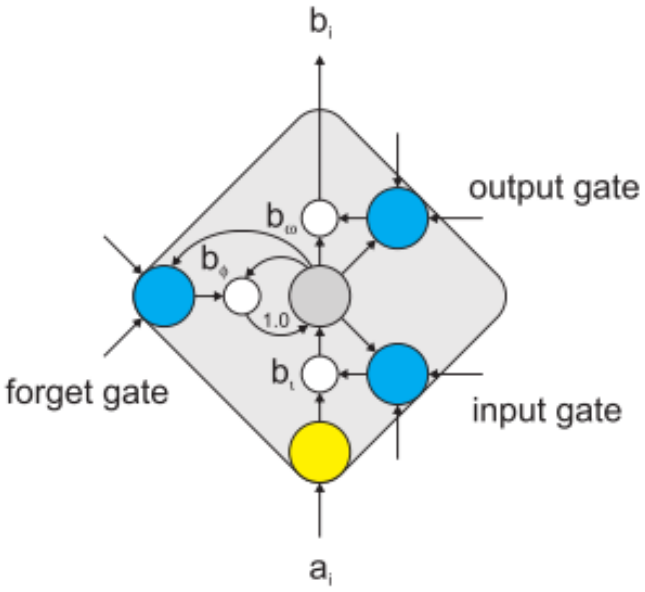


Figure 2.5 – An illustration of LSTM hidden unit. Figure borrowed from (Sundermeyer et al., 2012)

to one, hence preventing the gradient from vanishing or exploding. The element-wise non-linear f is replaced : each hidden unit is enriched by several so-called gated units (Figure 2.5). Sundermeyer et al. (2012) first introduce LSTMs to the language modelling task, and report an improvement of 8% in perplexity compared to a standard recurrent NNLM on English and French LM tasks, while also achieving considerable improvements in WER. LSTM recurrent NNLMs have also been applied to the lattice decoding problem in ASR, resulting in an improvement in WER up to 10.7% relative over a state-of-the-art baseline (Sundermeyer et al., 2014).

However, in spite of their benefits, recurrent NNLMs are still somewhat limited in their use due to expensive training and inference. The main computational bottleneck still lies between the hidden and output layer, just like the standard structure for which some solutions have been proposed (see Section 2.3.4 for a detailed description). By using the recurrent structure, this computational complexity is even harder to reduce, as the training and inference have to be carried out in a left-to-right fashion, without straightforward possibility to be performed in *mini-batch* mode, or to pre-compute some parts of the network to increase processing speed, or to group predictions sharing the same context during the inference. Some speed-ups and tricks have been proposed in (Mikolov et al., 2011b) where the data is first divided into paragraphs, filtered and then sorted; the most in-domain data is placed at the end of each epoch. Other solutions consist of simplifying the structure. For instance, the hidden layer can be decreased by simulating a maximum entropy model using a hash function on n-gram features (Mikolov et al., 2011b). Le et al. (2012b) propose an approximation of the recurrent architecture involving limited histories. While its performance is showed to be close to the recurrent NNLM, its simplified architecture makes the associated training procedure readily compatible with all speed-ups proposed for the standard feed-forward network, including mini-batch and resampling methods. The training time is showed to be divided by 8 by this *model-specific* approximation without significant loss in performance compared to the recurrent NNLM.

2.3.3 Ranking language models

We have presented so far the continuous-space models used to estimate word probabilities. This probabilistic estimate is required by the classical n-gram language models in which sentences are evaluated by their likelihood, which is in turn computed as a product of the probabilities of all of their composing words. Collobert and Weston (2008); Collobert et al. (2011) however propose another view according to which non-probabilistic CSMs should also be considered. The approach leads to the use of a unified neural network architecture, which can model several natural language processing tasks including the language modelling task, and which results in the so-called *ranking language model*. Instead of estimating a probability distribution, the proposed model aims only at ranking the likelihood of words to occur in a given context.

In addition to the use of continuous representation, Collobert and Weston (2008); Collobert et al. (2011) suggest some further ideas that are related to the work presented in this dissertation :

- The output of the ranking LM is not a probability distribution, and is not normalized. This property suggests a workaround for the computational problem described in Section 2.3.4. For the language model, the parameters are trained using the following *max-margin* criterion:

$$\max\left[0, 1 - g_{\boldsymbol{\theta}}(x) + g_{\boldsymbol{\theta}}(x')\right] \tag{2.16}$$

  where $g_{\boldsymbol{\theta}}$ denotes the network output, x and x' represent respectively an example from the training data and its (negative) counterpart generated from a sampling routine. Another training strategy uses NCE algorithm (*Noise Contrastive Estimation*, Section 3.2). The objective function (2.16) is also similar to the discriminative max-margin criterion that will be described and experimented in Chapters 5.

- Collobert and Weston (2008) emphasize the superiority of the *semi-supervised learning* scenario in improving the generalization power of the model in the absence of hand-engineered features. Indeed, all tasks listed in the paper use annotated data, except the language modelling task. In this situation, the language model is first trained to estimate the word continuous feature vectors from unlabelled training data; these shared representations are included in each task-specific model which is then fine-tuned for prediction tasks. This scenario is similar to the training method described in Chapter 5 where we initialize a partly supervised and discriminative training process from a CSTM that has been pre-trained with maximum-likelihood optimization.

2.3.4 Solutions for the output layer

All neural networks so far proposed for NNLMs are limited by their expensive computational cost during the training and inference. Early work on CSMs has pointed out that the bottleneck is situated at the output layer where we have to normalize scores over all vocabulary words (Bengio et al., 2003a; Schwenk and Gauvain, 2004). According

to (Schwenk and Gauvain, 2004), for the standard feed-forward structure, the number of floating point operations needed to predict an output word is (the reader is invited to refer to Table 2.1 for notations) :

$$((n-1) \times \mathrm{D} + 1) \times \mathrm{H} + (\mathrm{H} + 1) \times |\mathcal{V}^o| \tag{2.17}$$

The first term corresponds to the computation at the hidden layer, and the second term to the output layer. [4] In practice, we often have $|\mathcal{V}^o| \gg \mathrm{D}$ and $|\mathcal{V}^o| \gg \mathrm{H}$, therefore the second term, which corresponds to the processing at the output layer, represents most of the computational burden. With large-vocabulary applications where $|\mathcal{V}^o|$ is up to hundreds of thousand, this cost is problematic. Remarkably, the problem concerns the use of continuous-space language and translation models, no matter of how these models are incorporated in the MT application, either via N-best rescoring. (Schwenk et al., 2007; Kalchbrenner and Blunsom, 2013; Hu et al., 2014), direct decoding (Vaswani et al., 2013), or direct translation using neural network models [5] Model-specific workarounds often employ three main techniques : reducing the size of $\mathcal{V}^o$ (Bengio et al., 2003a; Schwenk and Gauvain, 2004; Schwenk et al., 2007), performing the normalization of (2.12) only on small groups of words (Morin and Bengio, 2005; Le et al., 2011, 2013), or not normalizing at all (Mnih and Teh, 2012; Vaswani et al., 2013; Xiao and Guo, 2013; Devlin et al., 2014). Besides that, application-specific solutions are also possible which exploit properties of the computing task implying the CSM, as described in the last part of this section. Moreover, some training criteria, such as those described in Chapter 5, do not require any normalization, either in training or in inference.

Short-list approach

As the computational cost of the output layer is proportional to the output vocabulary size $|\mathcal{V}^o|$, the most straightforward approach is to limit this quantity. For instance, Bengio et al. (2003a) propose to merge all low frequency words (whose frequencies ≤ 3) into one special token in the output layer, reducing $|\mathcal{V}^o|$ and resulting in 2-3 times speed-up without significant degradation of the performance. Furthermore, Schwenk and Gauvain (2004) propose to include in the output layer only the most frequent words, referred to as a short-list, and to use LM probabilities from a 4-gram back-off model for all the remaining words. More precisely, the probability of w_i, given its context $\mathbf{c}(w_i)$ is estimated as follows :

$$\mathbf{p}(w_i|\mathbf{c}(w_i)) = \begin{cases} \mathbf{p}_{\boldsymbol{\theta}}(w_i|\mathbf{c}(w_i)) \times \mathbf{p}_S(\mathbf{c}(w_i)) & \text{if } w_i \in \text{short-list} \\ \mathbf{p}_B(w_i|\mathbf{c}(w_i)) & \text{else} \end{cases}$$

where $\mathbf{p}_{\boldsymbol{\theta}}(.)$ is the probability estimated by the NNLM containing only the short-list, $\mathbf{p}_B(.)$ is from a back-off LM model, and the coefficient $\mathbf{p}_S(.)$ serves as a normalization constant ensuring that all the probabilities with context $\mathbf{c}(w_i)$ sum to 1.

In order to have any practical utility, several recent works on neural network models use short-lists which take up to 80K most frequent words (e.g (Sutskever et al., 2014)), which

[4] The input layer (also called the *lookup table* layer) consists of looking for columns of $\mathbf{C}$ corresponding to context words; with an efficient storing of $\mathbf{C}$, this operation can be assumed to have a constant cost.

[5] See for example a recent attempt in (Jean et al., 2015) dealing with the problem due to large vocabularies for Neural Machine Translation task.

is quite large. Short-lists can be limited to 2000 (Schwenk and Gauvain, 2004), however with the resort to classical discrete LMs. The necessity for a CSM to be supported by a back-off LM certainly hinders the full exploitation of its generalization capacity, as rare words outside the short-list are where the CSM should perform sharply better than its discrete counterpart, the final estimated scores however rely on the discrete model.

Hierarchically structured output vocabularies

A more promising approach is based on a hierarchical structuring of the output vocabulary (Morin and Bengio, 2005; Mnih and Hinton, 2008; Le et al., 2011). In its general form, all the elements from $\mathcal{V}^o$ are arranged into a tree structure. Each word is attached to a leaf of the tree, hence predicting output words is equivalent to the prediction of a path from the root of the tree to one of its leaves.

This idea is inspired from class-based LMs (Section 2.2.2) applied to CSMs. However, the neural structure allows us to go further by naturally considering word classes (or branches under each internal node), just like words themselves, as labels to be predicted using the context history $\mathbf{h}$. Indeed, by using the same neural structure either when predicting classes, or when predicting words within a class, we can incorporate context information in all our predictions. By this, the probability of a word within a class do not depend solely on this class (as in Equations (2.5), (2.6), (2.7), (2.8)), but also on the history. As for the computational complexity, the second term of (2.17) becomes :

$$(\mathrm{H}+1) \times \max_j\{S_j\} \times h$$

where $\max_j\{S_j\}$ denotes the maximum number of children at each internal nodes of the tree, and h is its height. This term is much smaller compared to the original one. Such a technique can result in 15 times speed-up at a small cost in accuracy (Mikolov et al., 2011b).

The remaining question is about the construction of such tree structure, and how to choose the structure leading to an optimal performance. Morin and Bengio (2005) suggest that it is possible to use an expert knowledge (WordNet), but the method degrades the performance. Another limitation is that such expert knowledge can be unavailable for certain domains or languages.

Considering the important impact of this hierarchical structure on the final performance, Mnih and Hinton (2008), Le et al. (2011), Le et al. (2013) propose to learn this structure from the word continuous representations, which will keep all the training phase independent from any external source information. A detailed description of this method will be given in Section 3.1.1 of Chapter 3.

Un-normalized and self-normalized output layers

The expensive cost of normalizing scores over the output vocabulary poses a serious question about the necessity of this step. Indeed, one may not perform such normalization,

α	$\log(\mathbf{p}(w\|\mathbf{c}))$	$\|\log \mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c})\|$
0	-1.82	5.02
10^{-2}	-1.81	1.35
10^{-1}	-1.83	0.68
1	-1.91	0.28

Table 2.2 – Normalization constants of the models trained in (Devlin et al., 2014), using the criterion (2.18). In the best case corresponding to $\alpha = 1$, $|\log(\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c}))|$ is reduced to 0.28, corresponding to a normalization constant of 1.32 or 0.76. The numbers are extracted from the same article.

as in the case of ranking LMs (Section 2.3.3). However, the lack of a probabilistic interpretation is problematic for the incorporation of such models into applications, such as in SMT. In order to understand the importance of such interpretation, we refer to Equation (2.1), where probabilities given to each word w_i are just an intermediate stage to proceed to an evaluation of hypotheses. The sentence-level score is the foundation of many useful applications of CSMs in Speech Recognition (Equation (2.3)) or in SMT where the system performance is evaluated on the whole sentence, not on the choice of each individual word. It it therefore necessary for LMs to be able to assemble the word probabilities in order to give a score to the entire output sentence. [6] Therefore, the work in (Collobert and Weston, 2008; Collobert et al., 2011) only provides a pre-training phase to obtain good word representation which can then be used in other prediction tasks.

In view of using NNLMs in SMT systems, it is necessary that the NN scores on all output words sum to 1. Let $\mathbf{a}_{\boldsymbol{\theta}}(w, \mathbf{c})$ be the activation corresponding to a word w given the context $\mathbf{c}$, this constraint translates as :

$$\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c}) = \sum_{w \in \mathcal{V}^o} \mathrm{e}^{\,\mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})} = 1$$

Devlin et al. (2014) present an attempt to make this condition explicit in the training criterion where $\log(\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c}))$ is optimized to be as close to 0 as possible. This is achieved by adding the term $\log^2(\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c}))$ to the objective function as follows :

$$\mathcal{L}_{devl}(\boldsymbol{\theta}) = \sum_{(w,\mathbf{c}) \in \mathcal{S}} \left[-\mathbf{a}_{\boldsymbol{\theta}}(w|\mathbf{c}) + \alpha \log^2(\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c})) \right] + \mathcal{R}(\boldsymbol{\theta}) \tag{2.18}$$

where the training set $\mathcal{S}$ is considered to be a set of *word-context* pairs $(w, \mathbf{c})$, and $\mathcal{R}(\boldsymbol{\theta})$ is a regularization term. With $\log(\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c}))$ made close to 0 during the training phase, the output scores become :

$$g_{\boldsymbol{\theta}}(w, \mathbf{c}) = \mathbf{a}_{\boldsymbol{\theta}}(w, \mathbf{c}) \tag{2.19}$$

which means that during the inference we do not need to normalize over the output vocabulary, but only to compute the output activations $\mathbf{a}_{\boldsymbol{\theta}}$. This self-normalization has resulted in about a 15 times speed-up in (Devlin et al., 2014). In Section 3.3.1 of Chapter 3,

[6] Being deprived of a probabilistic interpretation, we can no longer use Equation (2.1). Such situation requires a sentence-level training criterion (for instance in (Auli and Gao, 2014), or the strategies that will be described in Chapter 5), instead of a word-level criterion.

we present a similar result with a *self-normalization* network corresponding to a speed-up up to 50 times (Table 3.5). The difference in ratios may be due to the fact that the experiments in (Devlin et al., 2014) use quite modest vocabularies with only 32K words, while in the scope of this dissertation, experiments are always ran with large vocabularies of 500K words. [7]

The techniques of (Devlin et al., 2014) however do not speed up the training phase, as activations for all $w \in \mathcal{V}^o$ are still necessary in the training criterion (2.18). There exists a much faster self-normalization technique using the *Noise Contrastive Estimation* (or NCE) (Gutmann and Hyvärinen, 2010) and which has been applied to NNLMs (Mnih and Teh, 2012; Vaswani et al., 2013). Devlin argues that the NCE algorithm lacks a mechanism to control the degree of self-normalization, contrary to the presence of the hyper-parameter α in his objective function. However, this point seems to be practically irrelevant, as observed in our experiments in Chapter 3, the un-normalized scores from the NCE-trained CSMs, once incorporated in the SMT system, tend to approach the performance of the conventionally normalized NNLM without any significant loss in BLEU score. The quality of the normalization, as results of the NCE algorithm, seems to be sufficiently good to guarantee an integration of NCE-trained models using standard scenarios (Section 2.5.2). Table 2.2 shows some examples of the self-normalization process in (Devlin et al., 2014) controlled by the value of α.

Recently, Auli and Gao (2014) train a un-normalized model which results in 5 times faster inference. However, the model is not self-normalized, but is trained using a sentence-level objective function (Section 5.4.3).

Other speed-ups and tricks

Other tricks can be used to speed up the training and inference without any impact on performance; some of them are crucial in order to make the training of large-scale systems feasible. For instance, the training of NNLMs on very large corpora with billions of word tokens cannot be performed exhaustively, but requires to adopt *resampling* strategies, consisting of randomly selecting a small subset of training data to be used at each epoch. It has also been observed in (Schwenk, 2007) that data resampling can increase the generalization performance.

Speed-ups can also be obtained by propagating several examples at once throughout the network (Schwenk and Gauvain, 2004). This *mini-batch* mode allows us to use matrix-matrix operations, which are optimized by BLAS libraries to be faster than vector-matrix operations. [8] With 128 examples in each mini-batch, the training step is 10 times faster, though the modification does not yield any reduction on the number of floating point operations. In (Schwenk et al., 2012), it is showed that using optimized library for GPU makes the training and inference of NNLMs even faster.

[7] Fast matrix operations BLAS implemented in different libraries make the real-time comparisons only approximative, contrary to the comparison based on the number of floating point operations. For instance, the softmax proceeded on 500K words is *not* always 10 times slower than the one on 50K words.

[8] It is noticed that matrix-matrix operations are more easily accelerated on advanced CPU architectures with multi-threading.

However, it is important to notice that all the afore-mentioned tricks are much easier to be implemented with the feed-forward than other more complicated architectures. For recurrent NNLMs for instance, word predictions need to be processed in the left-to-right style, while the network's input must take into account the hidden vector computed at the preceding position. In such situation, the *mini-batch* mode can only be performed with the division of the data into b paragraphs (suppose b is the size of mini-batches), each propagation then takes the first non-processed word from each paragraph (Le et al., 2012b).

For the inference, a possible technique is *context grouping*, which consists of grouping n-grams sharing the same history words, then forwarding through the network each context only once. If the inference is carried out within the decoding of a SMT system (Vaswani et al., 2013; Devlin et al., 2014) (Section 2.5.2), the *pre-computation* of the hidden layer is showed to be very powerful with speed-ups up to 1000 times (Devlin et al., 2014), but applicable only on self-normalized, feed-forward neural architectures with one hidden layer. The idea is to pre-compute as most as possible Equation (2.10), in particular the dot product $\mathbf{W}^h \times \mathbf{i}$, and to store the results in memory before the decoding starts. The matrix $\mathbf{W}^h$ is divided to $n-1$ sub-matrices, each corresponds to a position in the context. Then, for every word from the input vocabulary, and for each position, the dot product between the word feature vector and the corresponding sub-matrix of $\mathbf{W}^h$ is computed and results are stored in $n-1$ lookup tables. During the decoding, the computation consists of simply looking to the right column of the right lookup table (corresponding to a position in the context). All $n-1$ found vectors are then summed up, along with the bias vector $\mathbf{b}^h$, before being fed to the non-linear function f . Hidden vectors can also be indexed to context sequences, then stored in memory cache in order to reduce as most as possible all costly operations during the decoding (Vaswani et al., 2013).

2.4 Continuous space translation models

The previous section has described various aspects of language models and the application of neural networks (or continuous-space models) in the language modelling task. In this section, we focus on the use of continuous-space models in modelling the translation process. Like language models which aim at estimating the probability of a word sequences $\mathbf{p}(w_1^N)$, translation models (TMs) are designed to evaluate the likelihood of a sequence $\mathbf{t} = t_1^I$, conditioned on another (foreign language) sequence $\mathbf{s} = s_1^J$. Translation models are hence by nature conditional distributions $\mathbf{p}(\mathbf{t}|\mathbf{s})$ [9], to which, as described in Chapter 1, we add latent variables $\mathbf{a}$ which model intermediary stages, such as the translation of each source word or sequence of source words, or the reordering of local translations in order to form a complete meaningful sentence. A probability distribution on target sentences given the source sentence is defined as :

$$\mathbf{p}(\mathbf{t}, \mathbf{a}|\mathbf{s}) = \frac{1}{\mathrm{H}(\mathbf{s})} \exp\left(\sum_{m=1}^{M} \lambda_m f_m(\mathbf{s}, \mathbf{t}, \mathbf{a})\right) \tag{2.20}$$

[9] In practice phrase-based SMT systems include phrase translation probabilities in both directions, i.e $\mathbf{p}(\mathbf{t}|\mathbf{s})$ and $\mathbf{p}(\mathbf{s}|\mathbf{t})$.

where M feature functions (f_m) are weighted by a set of coefficients $\boldsymbol{\lambda} = \lambda_1^M$, and $\mathrm{H}(\mathbf{s})$ is the normalization constant. In most cases, this set of feature functions contains the translation models delivering a score for each derivation $(\mathbf{t}, \mathbf{a})$.

Like in the case of discrete language models, early translation models relied on maximum likelihood estimates of discrete random variables and exploit relative frequencies from the training data. These models suffer from the *data sparsity* problem, as the values of these variables are considered without any relationship between them, while count-based estimations are in many cases not statistically reliable. In the case of TMs, the problem becomes even more severe, as discrete variables take values from sets of word or phrase pairs, which are larger than the word vocabulary; whereas training data for bilingual TMs is not as abundant as the one used for LMs. Remedies exist, such as smoothing techniques (Section 2.2.1) [10], or resorting to *factored language models* (Bilmes and Kirchhoff, 2003) adapted to TMs in (Koehn and Hoang, 2007; Crego and Yvon, 2010). However, such approaches necessitate external linguistic analysis tools which are not always available; moreover they do not seem to bring consistent improvements.

Continuous-space models (CSMs), as described in Section 2.3 seem to be a promising approach to improve the estimation of these models. In many cases, continuous-space translation models (CSTMs) are a bilingual generalization of LMs, in which words come from two different languages, instead of only one. Such models are also often called *Neural Network Joint Models* (NNJM) (Devlin et al., 2014) as they can be employed to estimate monolingual as well as bilingual probabilities. Various structures and training strategies that have been used for NNLMs can be generalized to CSTMs. We can divide these models into *Phrase-pair based* TMs and *word factored* (or lexicalized) TMs; the first group manipulates phrase pairs as elementary translation units, while the second one factors probabilities at the level of individual words, hence amounts to a *word based* LM-style model.

2.4.1 Phrase- and phrase-pair- based CSTMs

In the context of SMT, Schwenk et al. (2007) present the first CSTM estimating translation probabilities, more precisely within the framework of n-gram-based SMT system. This framework differs from other approaches by the use of latent variables indicating the reordering of the source sentence, the segmentation of this source into segments, and the translation of each segment forming bilingual translation units called *tuples* (Section 2.4.2). The translation model estimates the joint probability $\mathbf{p}(\mathbf{t}, \mathbf{s}|\mathbf{a})$ by first decomposing it at the level of tuples :

$$\mathbf{p}(\mathbf{t}, \mathbf{s}|\mathbf{a}) = \mathbf{p}(\mathrm{u}_1, ..., \mathrm{u}_L) = \prod_{l=1}^{L} \mathbf{p}(\mathrm{u}_l|\mathrm{u}_{l-n+1}^{l-1})$$

where $\mathrm{u}_1, ..., \mathrm{u}_L$ are L phrase pairs (tuples) extracted from $(\mathbf{t}, \mathbf{a}, \mathbf{s})$, and u_{l-n+1}^{l-1} denotes the context using the n-gram assumption. Inspired from the NNLM, each tuple can be

[10] These methods have been systematically compared in a phrase-based SMT system (Foster et al., 2006).

projected into a multi-dimensional continuous space, in which similarities can be computed as indications of relationship between different tuples. The neural structure used in that work is still the feed-forward network, while in order to reduce the computational cost, the output vocabulary is limited to 8K most frequent units. This short-list approach (Section 2.3.4) necessitates to interpolate the CSTM with a back-off LM on tuples. The CSTM is then used to rescore 1000-best lists (see Section 2.5.2 for more details). The results only show a slight improvement of 0.2 BLEU points [11], whereas the CSTM seems to under-perform the word-based NNLM which has been used to rescore the same N-best lists. This results is particularly surprising, as unlike the NNLM, the CSTM explicitly exploit information about the source language in context as well as in prediction. It suggests that estimation problems are still present despite the use of the continuous representation of phrase pairs. Another limitation of that work is the *Short-list* approach which makes the model applicable only on small-scale tasks with limited number of phrase pairs (about 21K units as considered in the paper).

Zamora-Martinez et al. (2010) propose a tighter integration of a CSM within an n-gram-based SMT system both for the phrase-pair based CSTM and the target NNLM. Later works, such as (Hu et al., 2014) try to use more complicated neural structures, such as the recurrent neural network for CSTMs. The main motivation, like in the case of recurrent NNLMs, is to handle longer dependencies between (source and target) contexts and the unit to be predicted. However recurrent neural network models on phrase pairs face a more severe problem of data sparsity, as the set of phrase pairs is much larger than the set of words. Phrase pairs are modelled as *Minimum Translation Units* (MTUs) in (Hu et al., 2014), which correspond to a segmentation of the source and target sentences satisfying two constraints : there are no overlapping word alignment links between phrase pairs, moreover it is impossible to extract smaller *translation units* without violating the word alignment constraint. The translation is then modelled as a sequence of MTUs just like in the n-gram model of (Schwenk et al., 2007), however using a recurrent neural network exploiting *unbounded* contexts. The CSTM is used to rescore N-best lists, and gives quite limited improvements. Remarkably, the result confirms the trend observed in (Schwenk et al., 2007) according to which the atomic MTU-based model hinders the MT performance compared to a word-based recurrent NNLM (-0.7 BLEU point difference). The authors propose a workaround by considering the MTU not as an atomic unit, but as a *bag-of-words*, and redesigning the output layer to predict a word rather than a MTU. This modification helps to bring the performance of the MTU-based model close (but always inferior) to the recurrent NNLM. The *bag-of-words* representation seems to hinder the model performance, as dependencies between words from different MTUs are not explicitly modelled. The capacity of handling longer contexts which derives from the modelling of MTUs seems to have little importance. Finally, Zhang et al. (2015) represent another attempt to generate the target sentence phrase by phrase, by the definition of *Minimal Phrase* in the context of a bilingual sentence pair model. The proposed model is however a target NNLM on phrases without any conditioning on source contexts; moreover phrase-based models need to be interpolated with word-based models to give improved results.

[11] see Section 1.2 for more details on BLEU.

2.4.2 Word-factored n-gram-based CSTM

The development of CSTMs has a history which is somewhat contrary to what has been observed in SMT. For the development of SMT systems, the mainstream has been observed to shift from word-based models (Brown et al., 1993) to phrase-based approach (Zens et al., 2002) which has been a significant progress, as atomic-phrase consideration allows the model to enlarge the context information from both source and target words. However, the performance of phrase-based CSTMs is hindered by a huge number of events to be taken into account by the neural network, which increases drastically the number of free parameters and the model complexity, whereas training data is reduced. Due to this estimation problem, the most successful CSTMs use words, instead of phrases or phrase pairs, as minimal units. The goal is hence to predict (source and target) words from a certain history made of context words, or to evaluate a translation based on its constituting words.

The n-gram-based CSTMs described in (Le et al., 2012a) clearly reflect this trend. In their first proposal, the authors reuse the idea of building an n-gram CSTM over the set of phrase pairs (or *tuples*), using a hierarchically structured output layer (Section 2.3.4) to handle large vocabularies of tuples or phrases, whereas the context length is increased to 10-gram instead of only 3-gram in (Schwenk et al., 2007). The number of all phrase pairs is quite impressive : 300K for a small-scale task and 8847K for a large-scale one, setting this work as one of the first large-scale experiments on CSTMs. The results confirm the trend observed in (Schwenk et al., 2007) : the phrase pair based CSTM improves the baseline system only by 0.5 BLEU points via N-best rescoring on the small task, while a target NNLM on top of the same baseline gives 1.1 BLEU point improvement. Further investigating on this comparison, the authors propose to compare the phrase pair based CSTM with phrase-factored and word-factored (or lexicalized) models, the results show that decomposing phrase pairs into smaller units yields significant improvements during the N-best rescoring. Finally, only the word-factored CSTM outperforms the target NNLM (+0.6 BLEU points). This comparison suggests that, even though the continuous representation of atomic units (phrase pairs, phrases, or words) has been used and a structured output layer allows us to handle all units, estimation problems persist, and it is important to factorize the probabilistic model in order to limit the vocabulary size.

The CSTMs used in the scope of this dissertation are based on the word-factored model introduced in (Le et al., 2012a), it is hence important to give a detailed description of different formulations leading to this structure. In spite of the proposition of various neural structures in the literature, we still advocate the conventional feed-forward structure. The main reason is that powerful speed-ups are available for both its training and inference (Section 2.3.4). Moreover, simple feed-forward networks still achieve impressive improvements (Le et al., 2011, 2012a; Allauzen et al., 2013; Devlin et al., 2014; Do et al., 2014a).

The model is based on the n-gram approach for SMT (Section 1.1.3) and has been first described in (Le et al., 2012a). In this approach, the sentence pair is assumed to be decomposed into a sequence of L bilingual phrase pairs (called *tuples*) defining a joint segmentation $(\mathbf{s}, \mathbf{t}) = (\mathrm{u}_1, ..., \mathrm{u}_L)$ (see Figure 1.1). The basic translation units are *tuples*,

representing a matching $\mathrm{u} = (\overline{s}, \overline{t})$ between a source $\overline{s}$ and a target $\overline{t}$ phrase. The first formulation of the model comes from the n-gram assumption over sequences of tuples :

$$\mathbf{p}(\mathbf{s}, \mathbf{t}) = \prod_{l=1}^{L} \mathbf{p}(\mathrm{u}_l | \mathrm{u}_{l-n+1}^{l-1}) \tag{2.21}$$

All the probabilities in Equation (2.21) are modelled by a feed-forward neural network, which is referred to as the standard n-gram CSTM in (Le et al., 2012a). The elementary units are bilingual phrase pairs, which means that the underlying vocabulary, hence the number of parameters, can be quite large, even for small translation tasks. In order to reduce the number of free parameters, Equation (2.21) can be factored by decomposing tuples in two (source and target) parts and in two equivalent ways :

$$\begin{aligned} \mathbf{p}(\mathrm{u}_l | \mathrm{u}_{l-n+1}^{l-1}) &= \mathbf{p}(\overline{t}_l | \overline{s}_{l-n+1}^{l}, \overline{t}_{l-n+1}^{l-1}) \times \mathbf{p}(\overline{s}_l | \overline{s}_{l-n+1}^{l-1}, \overline{t}_{l-n+1}^{l-1}) \\ &= \mathbf{p}(\overline{s}_l | \overline{s}_{l-n+1}^{l-1}, \overline{t}_{l-n+1}^{l}) \times \mathbf{p}(\overline{t}_l | \overline{s}_{l-n+1}^{l-1}, \overline{t}_{l-n+1}^{l-1}) \end{aligned} \tag{2.22}$$

Each decomposition involves two bilingual conditional distributions, the first represents a TM, the second term is best viewed as a distortion model. Another benefit of this formulation (referred to as the factored n-gram CSTM in (Le et al., 2012a)) is that elementary events now correspond either to source or target phrases, but not to phrase pairs. The underlying vocabulary is thus obtained as the union, rather than the cross product, of phrase inventories.

Finally, the third formulation (the word-factored CSTM in (Le et al., 2012a)) consists of taking (source and target) words as the basic units of the n-gram TM. This decomposition of phrases in words is considered only as a way to mitigate the parameter estimation problem. Indeed, as the baseline system (hence the corresponding decoder) is still n-gram-based, the neural network still computes, as its final outputs, the probabilities of phrases rather than of words. Let s_l^k denote the k^{th} word of the source sequence $\overline{s}_l$, and t_l^k denote the l^{th} word of the target sequence $\overline{t}_l$. We also denote $\mathbf{c}_{n-1}(s_l^k)$ (respectively $\mathbf{c}_{n-1}(t_l^k)$) the sequence made of the $n-1$ words preceding s_l^k (resp. t_l^k) in the source (resp. target) sentence. [12] The joint probability can then be rewritten in two equivalent ways, as :

$$\begin{aligned} \mathbf{p}(\mathbf{s}, \mathbf{t}) &= \prod_{l=1}^{L} \left[\prod_{k=1}^{|\overline{t}_l|} \mathbf{p}(t_l^k | \mathbf{c}_{n-1}(t_l^k), \mathbf{c}_{n-1}(s_{l+1}^1)) \times \prod_{k=1}^{|\overline{s}_l|} \mathbf{p}(s_l^k | \mathbf{c}_{n-1}(t_l^1), \mathbf{c}_{n-1}(s_l^k)) \right] \\ &= \prod_{l=1}^{L} \left[\prod_{k=1}^{|\overline{s}_l|} \mathbf{p}(s_l^k | \mathbf{c}_{n-1}(s_l^k), \mathbf{c}_{n-1}(t_{l+1}^1)) \times \prod_{k=1}^{|\overline{t}_l|} \mathbf{p}(t_l^k | \mathbf{c}_{n-1}(s_l^1), \mathbf{c}_{n-1}(t_l^k)) \right] \end{aligned} \tag{2.23}$$

This decomposition relies on the n-gram assumption at the level of words, rather than phrases or phrase pairs. Intuitively, the conditioning part includes two sliding windows of length $n-1$, one for each language; however the moves of these windows remain synchronized by the source reordering and the segmentation into tuples. Moreover, the context is not limited to the current phrase, but continues to include words in adjacent

[12] The source sentence is reordered beforehand to match the target sentence word order. See Section 1.1.3 for more details.

$$P(\bar{s}_{11}: \text{nobel de la paix} \mid \bar{t}_{11}: \text{nobel peace}, \bar{s}_9: \text{recevoir}, \bar{t}_9: \text{receive}, \bar{s}_{10}: \text{le}, \bar{t}_{10}: \text{the}) = \frac{P(\bar{t}_{11}: \text{nobel peace} \mid \bar{s}_{11}: \text{nobel de la paix}, \bar{s}_9: \text{recevoir}, \bar{s}_{10}: \text{le}, \bar{t}_9: \text{receive}, \bar{t}_{10}: \text{the})}{P(\bar{s}_{11}: \text{nobel de la paix} \mid \bar{s}_9: \text{recevoir}, \bar{s}_{10}: \text{le}, \bar{t}_9: \text{receive}, \bar{t}_{10}: \text{the})}$$

$$= P(\text{nobel} \mid \text{recevoir, le}, \text{receive, the})$$
$$\times P(\text{de} \mid \text{le, nobel}, \text{receive, the})$$
$$\times P(\text{la} \mid \text{nobel, de}, \text{receive, the})$$
$$\times P(\text{paix} \mid \text{de, la}, \text{receive, the})$$

$$\times P(\text{nobel} \mid \text{la, paix}, \text{receive, the})$$
$$\times P(\text{peace} \mid \text{la, paix}, \text{the, nobel})$$

Figure 2.6 – An example of the decomposition of the tuple-based model at the level of phrases, then at the level of words. In the first formulation, tuples are treated as the smallest translation units, while in the second one, each tuple is decomposed into its source and target side. The third formulation estimates the probability of words, given the context composed of the two most recent source and target words (for a trigram model). Source words are coloured in red, while target words are coloured in green.

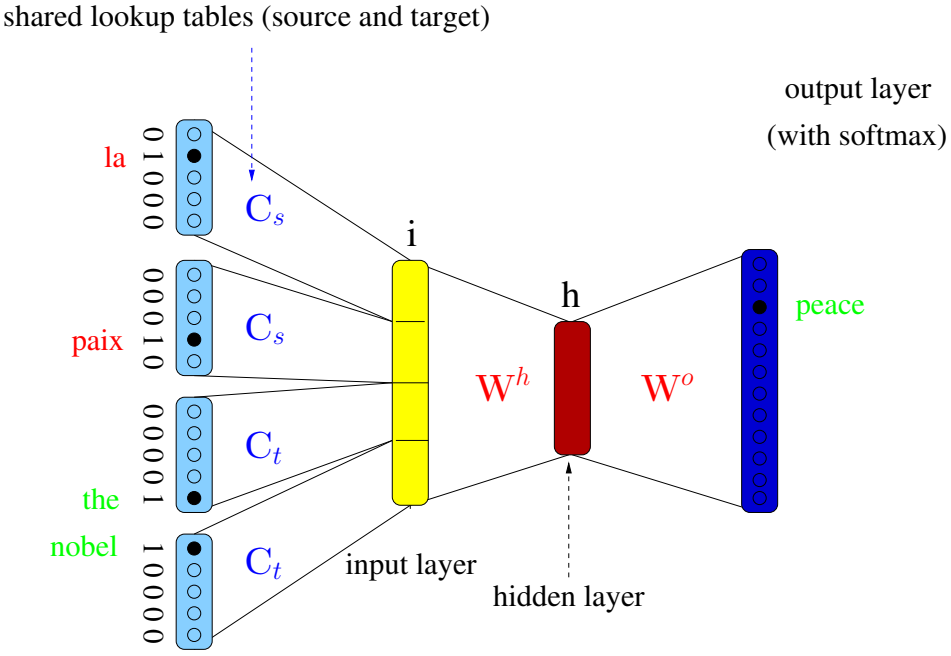


Figure 2.7 – The feed-forward neural network model used to estimate the last word-factored probability in the example of Figure 2.6.

phrases. Figure 2.6 gives an example of these three formulations applied on a bilingual trigram model.

As with for the SOUL NNLM (Le et al., 2011, 2013), each formulation described by Equations (2.21), (2.22) and (2.23) uses a neural network model with structured output layer (Section 2.3.4) to compute the output probabilities. Figure 2.7 illustrates the neural architecture used to estimate the word-factored probabilities given by the last formulation. A major modification compared to the standard NNLM lies in the input layer where the CSTM uses two mapping functions, one for each (source and target) language. These three formulations have been compared in (Le et al., 2012a) on an English-French translation task within the framework of N-best list rescoring (Section 2.5.2). The results show that in spite of the presence of source words which are directly aligned to the target predicted word throughout the tuple segmentation and provide strong suggestion about which word to be present, only the third formulation outperforms the target NNLM (+0.6 BLEU points). The atomic representation of phrases or phrase pairs seems to be harmful to the general performance. Henceforth, in the scope of this dissertation, we will only use the word-factored formulation, expressed by Equation (2.23) in all experiments.

2.4.3 Other lexicalized CSTMs

In order to predict words, most of lexicalized CSTMs use a softmax layer covering all vocabulary words. With large-vocabulary SMT systems, CSTMs face the same computational problems as the NNLMs, hence all strategies described in Section 2.3.4 for NNLMs can be straightforwardly applied to CSTMs, for instance the *Structured OUtput Layer* (Le et al., 2012a), the self-normalized output layer in (Devlin et al., 2014) or the NCE algorithm used to train CSTMs described later in Chapter 3.

Besides the output layer, the main divergence lies in the input layer where different strategies of context handling have been proposed. Being the first work on lexicalized CSTM, Le et al. (2012a) employ a context composed of an $n-1$-gram sliding window of target words concatenated with a similar window of source words, while parameters are organised in the same way as in the standard NNLM (Bengio et al., 2001). This context handling is inspired (and deduced) from the n-gram-based approach in SMT : source and target windows are synchronised by the segmentation of source and target sentence into phrase pairs whereas the source side is reordered beforehand. Also feeding $n-1$ previous target words into the history, Devlin et al. (2014) however propose a different way of choosing the sliding window of source context words (denoted by $\mathcal{S}_j$, where j is the current target word to be predicted). This window is chosen as being the most relevant to the current target position, established through the word alignments. Each target word t_j is supposed to be *affiliated* with exactly one source word at index a_j, then $\mathcal{S}_j$ comprises the m words surrounding s_{a_j} :

$$\mathcal{S}_j = s_{a_j - \frac{m-1}{2}}, ..., s_{a_j}, ..., s_{a_j + \frac{m-1}{2}} \tag{2.24}$$

The notion of affiliation is directly derived from the word alignment, but unlike the synchronised contexts in (Le et al., 2012a), the context described in (Devlin et al., 2014) does not depend on the reordering of the source sentence, as well as the segmentation

into phrase pairs. This *purely lexicalized* model is reported to have good performance. [13] Note however that a combination of several CSTMs is used in this paper. [14]

Also focussing on the incorporation of source context words in a CSTM by using a window covering all surrounding words as defined in Equation (2.24), Tran et al. (2014) suggest another use of lexicalized models in improving the prediction accuracy for word translations when translating into a morphologically rich language. A peculiarity of models described in this paper lies in the fact that word-based probabilities are further decomposed into the probabilities of stems and suffixes, resulting in two different tasks of predicting separately target stem and target suffix. The improved capacity in predicting morphological forms is then proved to be beneficial for a phrase-based SMT system from English into Russian. The differences between this model and those considered in this dissertation is that it predicts target words independently from one another. Moreover, instead of predicting words in a large output vocabulary, the model includes only a limited number of options deduced from the word alignment on training corpora; this feature allows the model to prevent some computational difficulties described earlier.

From context-dependent to source-dependent recurrent neural networks

Recurrent neural network structures have also been used in CSTMs, where the conditioning on words from the source language is an important feature which distinguishes these models from each other, and from the recurrent target language NNLMs. In order to further incorporate the source side information, the *auxiliary input vector*, inspired from *context-dependent* recurrent NNLMs (Section 2.3.2), plays an important role. This additional input vector contains the source sentence's embedding, which is combined with the recurrent hidden state and the precedent word t_{j-1}'s embedding, similarly to the mechanism illustrated in Figure 2.4. The only difference is that, instead of the topic-dependent vector as proposed in (Mikolov and Zweig, 2012), $\mathbf{f}_j$ here gives information about the source sentence which is useful for the prediction of t_j. Auli et al. (2013) provide a typical work along this line, where the source $\mathbf{s}$ is represented either by a *latent semantic analysis*, or by a *word encoding* obtained by training a recurrent NNLM on the source side. Further developments of that work use a lattice rescoring scenario to incorporate the CSTM into a SMT system. However, the unbounded recurrent context is not compatible with the recombination of decoder states sharing the same context into a single state (Koehn et al., 2007), as the amount of recombination decreases significantly with long contexts. The difficulty makes some simplifications necessary; but even in this case lattice rescoring does not seem to be significantly better than N-best rescoring. The integration leads to to an improvement of 1.5 BLEU points over the WMT 2012 French-English baseline, but the extension of the recurrent architecture brings only 0.2 BLEU points. This may be due to the difficulties in training recurrent neural networks (Section 2.3.2).

Another way to build the continuous representation of the source sentence is proposed in (Kalchbrenner and Blunsom, 2013) where the authors use a convolutional network to transform the whole source sentence into a multi-dimensional embedding (the *Convolutional Sentence Model* (Blunsom et al., 2013)). The prediction of each target word t_j

[13] +3.0 BLEU points on top of a powerful baseline already including a recurrent NNLM.
[14] 4 joint models and 2 lexical models with only source or target words in the conditioning part.

is conditioned on the source sentence by simply feeding this embedding to the auxiliary input vector $\mathbf{f}_j$. However, this model, as well as the model presented in (Auli et al., 2013) might have a weakness that the same source language context is used for the prediction of all words of the target sentence. As a consequence, Kalchbrenner and Blunsom (2013) propose a second model to make this conditioning on the source side information more precise. The motivation behind this is the fact that target words should depend more strongly on certain parts of the source sentence. It is hence likely that the prediction of each target word t_j should focus on some relevant context source words, such as those covered by the sliding window surrounding the *affiliated* position in (Devlin et al., 2014; Tran et al., 2014). The proposed model, by a mechanism called *Inverted Convolutional n-Gram Model*, or *ICGM*, first tries to generate I different vectors representing the same source words (I is the length of the target sentence), each vector is supposed to contain the necessary information for the word prediction at the corresponding target position. This variant of the recurrent CSTM yields indeed significant improvements in terms of perplexity [15] which is 40% lower than the original model. This suggests the importance of context source words in the conditioning part. More remarkably, instead of relying on word and phrasal alignments as in (Le et al., 2012a; Devlin et al., 2014), this work proposes the ICGM to automatically learn these dependencies from the training data. It would be interesting in a future work to repeat the English-French experiments described in the paper on other language pairs in order to see if such mechanism is capable of capturing long-span reordering. The idea of automatically learning word alignment using neural networks has also been pursued in (Bahdanau et al., 2014) with the so-called attention-based approach.

Generating translations with CSTMs

A consequence of an independence from word or phrasal alignments is that translations can be generated and evaluated without any baseline SMT system. This gives ways to a new research direction in which neural networks are used to directly generate translations from the source, without resorting to any linguistic information or hand-engineered features (Hermann and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014). The main idea can be described as first encoding the source sentence into a continuous feature vector, then using it as a part of the input to generate the target sentence, often within a recurrent NNLM incorporating an auxiliary input layer. Sutskever et al. (2014) propose to use a recurrent NNLM with Long Short-Term Memory (LSTM) (Section 2.3.2) hidden layer to encode the source sentence, then another NNLM to decode the target sentence. The direct translation using this model results in an encouraging BLEU score of 34.8 on WMT' 14 English-to-French task, compared to 33.3 obtained by a phrase-based SMT system, even though only a short-list of 80K most frequent words are used to generate the translations. The combination of these two systems through 1000-best rescoring helps to improve the baseline by 3.2 BLEU points. Similarly, Cho et al. (2014) use a recurrent encoder-decoder with a fixed-length vector representation for the whole source sentence, and a simpler variant of LSTM for their hidden layers.

In line with the modification brought out by the second model proposed in (Blunsom et al., 2013), Bahdanau et al. (2014) train a model that can automatically learn a *soft* word

[15] with respect to reference translations.

alignment evaluating the importance of each source word towards each target position. The auxiliary input vector $\mathbf{f}_j$ still has a fixed length, but is estimated from a set of vectors $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_J$ (J is the length of the source sentence). Each vector $\mathbf{h}_i$ represents the whole source sentence, but with strong focus on the part surrounding the i-th source word. These vectors are obtained from two recurrent NNLMs computed on the source sentence, one predicting in the left-to-right order and the other in the inverse order. Furthermore, as a counterpart to the *ICGM* proposed in (Kalchbrenner and Blunsom, 2013), the authors propose an additional feed-forward neural network to predict the impact of each source word s_i on the target word t_j. More precisely, the auxiliary input vector $\mathbf{f}_j$ is computed as a linear combination :

$$\mathbf{f}_j = \sum_{i=1}^{J} \alpha_{ij} \mathbf{h}_i$$

where $\alpha_{ij}, i = 1, .., J, j = 1, .., I$ are the outputs (computed from a softmax layer) of the additional neural network, which can be interpreted as the probability of the target word t_j being aligned to the source word s_i. Therefore, the model does not depend on a *hard* word alignment, but implicitly learns a *soft* alignment between source and target words from the training data.

Finally, we can remark that in most of past work on CSTMs, the best performances are often achieved by combining different models (4-6 models), or the proposed models are combined with baselines. Even though this strategy is necessary to achieve a good performance, the effect of training each individual model is not well assessed. In this dissertation, we insist particularly on the one-model configuration in order to present a fair framework in which various training strategies can be compared to each other.

2.4.4 Discriminative phrase translation models

We also review a particular kind of TMs which do not aim at estimating probabilities, but only a score reflecting the likelihood of a translation unit, or the *compatibility* between a source and a target phrase (Gao and He, 2013). More precisely, in the phrase-based SMT framework, the translation model, also known as *phrase table*, is often constructed by a two-step approach : first, phrase pairs are heuristically extracted from an automatically word-aligned training data; the second step consists in estimating a score for each phrase pair. These scores are conventionally based on relative frequencies estimated on the same aligned data (Koehn et al., 2003). This estimation procedure faces *data sparsity* problem, just like the discrete LMs whose probabilities have to be estimated on the occurrences of word sequences.

Therefore, most work on phrase translation models focusses on improving the parameter estimation procedure. He and Deng (2012) propose a new discriminative training procedure for phrase and lexicon translation scores, instead of relying solely on relative frequencies. The general training scenario is a joint optimization for $\{\boldsymbol{\theta}, \boldsymbol{\lambda}\}$ (Section 1.3.2), where $\boldsymbol{\theta}$ denotes the parameters of phrase and lexicon TMs, and $\boldsymbol{\lambda}$ represents the log-linear coefficients in Equation (2.20). The whole procedure is sketched in Figure 2.8. The parameters $\boldsymbol{\theta}$ are trained on the training set, while the feature weights $\boldsymbol{\lambda}$ are tuned alternatively on a development set. Due to possible mismatch between the training and

1. Build the baseline system, estimate $\{ \boldsymbol{\theta}, \boldsymbol{\lambda} \}$.
2. Decode N-best list for training corpus using the baseline system, compute $BLEU(E_n, E_n^*)$.
3. set $\boldsymbol{\theta}' = \boldsymbol{\theta}$, $\boldsymbol{\lambda}' = \boldsymbol{\lambda}$.
4. Max expected BLEU training
   a. Go through the training set.
      i. Compute $P_{\boldsymbol{\theta}'}(E_n|F_n)$ and $U_n(\boldsymbol{\theta}')$.
      ii. Accumulate statistics $\{\gamma\}$.
   b. Update: $\boldsymbol{\theta}' \rightarrow \boldsymbol{\theta}$ by one iteration of GT.
5. MERT on the tuning set: $\boldsymbol{\lambda}' \rightarrow \boldsymbol{\lambda}$.
6. Test on the validation set using $\{ \boldsymbol{\theta}, \boldsymbol{\lambda} \}$.
7. Go to step 3 unless training converges or reaches a certain number of iterations.
8. Pick the best $\{ \boldsymbol{\theta}, \boldsymbol{\lambda} \}$ on the validation set.

Figure 2.8 – Discriminative training procedure proposed in (He and Deng, 2012)

development set, the training process might not help to improve the BLEU score as expected. As a consequence, the authors propose to use another held-out data to validate $\{\boldsymbol{\theta}, \boldsymbol{\lambda}\}$: only the iteration which maximizes the training criterion on this data is finally chosen.

The objective function used in (He and Deng, 2012) is the *expected* BLEU score with Kullback-Leibler divergence regularization. This criterion, described in Section 1.3, has been originally used to train log-linear coefficients $\boldsymbol{\lambda}$ on a development set (Rosti et al., 2011). The main motivation behind this is the fact that the conventional maximum-likelihood estimation is not directly related to the translation performance, measured for instance by BLEU (Section 1.2). Gao and He (2013) propose a simpler optimization procedure which updates $\boldsymbol{\theta}$ based on Stochastic Gradient Descent, while the phrase translation probability is modelled using *Markov Random Fields* (MRF). Another difference is that the work in (Gao and He, 2013) does not use held-out data to validate $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$, but base on the BLEU scores on the development set to choose the best iteration at the end.

A continuous-space version of the model in (Gao and He, 2013) is proposed in (Gao et al., 2014), where each (source and target) phrase is projected into a (common) multi-dimensional continuous space. For each phrase pair, a score is used to indicate the *compatibility* between source and target phrases, and is computed as the distance between their embeddings. The model is still trained by the expected BLEU objective function, similarly to the algorithm presented in Figure 2.8. [16] However, the main weakness of this model is that phrase translation probabilities depend solely on its phrase embeddings, and not on other context words which fall outside phrase boundaries.

Recently, an attempt has been made to use the expected BLEU criterion to train a recurrent NNLM (Auli and Gao, 2014), which improves the system by 0.6 BLEU points compared to the same model trained with maximum-likelihood estimation. A peculiarity of that work is that the neural network does not employ a softmax layer, but delivers *un-normalized* scores. This feature makes the model in (Auli and Gao, 2014) similar to ranking LMs (Section 2.3.3). However, instead of using a *word-level* objective function (Equation (2.16)), the model is trained by a *sentence-level* criterion which is the expected

[16] The CSM is shown to improve a WMT'12 French-to-English system by up to 1.3 BLEU points.

BLEU. Moreover, as no normalization is required, the inference is reported to be 5 times faster than the conventional models, in spite of a slower training.

2.5 Evaluating and training a continuous space model

In this section, we discuss some aspects about evaluation and training strategies for CSMs. In general, it is desirable to have a training method that is consistent with the evaluation; however it is not always the case, for instance with language models. In practice, LMs are trained to optimize the perplexity, but then the same models are used in different applications for which the evaluation metric is only loosely related to the perplexity. Evaluation within these applications is often expensive as it requires the existence of a baseline system, along with a set of models corresponding to various system features. The evaluation of LMs is hence often performed separately. Indeed, during the training, it is often convenient to have an intermediary intrinsic evaluation measure that does not depend on any external system.

Since our work focusses on CSTMs, as extrinsic metrics, we only consider the integration of NNLMs and CSTMs into a SMT system, and consider the BLEU score to assess the final performance. Other evaluation methods exist, such as including NNLM into an ASR system, and using the *Word Error Rate* (WER) of the final system to assess the quality of the included LM.

2.5.1 Perplexity

For NNLMs and several CSTMs, the most straightforward intrinsic measure is the perplexity. This quantity is derived from the probabilistic interpretation formulated in Equation (2.1) : the model perplexity computed on a dataset $\mathcal{D}$ reflects the likelihood of $\mathcal{D}$ estimated by this model. Suppose that $\mathcal{D}$ contains d samples, $\mathcal{D} = \{\mathcal{D}_1, ...\mathcal{D}_d\}$, then its log-likelihood estimated by the model with parameters $\boldsymbol{\theta}$ is computed as :

$$\log \mathbf{p}(\mathcal{D}|\boldsymbol{\theta}) = \sum_{i=1}^{d} \log \mathbf{p}_{\boldsymbol{\theta}}(\mathcal{D}_i)$$

While evaluating a LM (or a word-factored TM), each data sample $\mathcal{D}_i$ is considered as the pair of context words and a word to be predicted in this context. In order to have a fair comparison that is data size-independent, the averaged log-likelihood is often used :

$$\text{average-log-likelihood}(\mathcal{D}|\boldsymbol{\theta}) = \frac{1}{d} \log \mathbf{p}(\mathcal{D}|\boldsymbol{\theta}) = \frac{1}{d} \sum_{i=1}^{d} \log \mathbf{p}_{\boldsymbol{\theta}}(\mathcal{D}_i) \tag{2.25}$$

The perplexity is computed from this quantity as :

$$\text{perplexity}(\mathcal{D}, \boldsymbol{\theta}) = \exp\left(-\frac{1}{d} \sum_{i=1}^{d} \log \mathbf{p}_{\boldsymbol{\theta}}(\mathcal{D}_i) \right) \tag{2.26}$$

which is used to evaluate the language or translation model with parameters $\boldsymbol{\theta}$, where $\mathcal{D}$ is not used during the training phase. In terms of the word prediction problem modelled by a LM, a lower perplexity computed on a held-out validation set implies a higher likelihood of the unseen data, which means that the model predicts better the new data samples.

The quantity (2.25) can also be viewed as a negative empirical estimate of the *cross-entropy* of the true (but unknown) data distribution $\mathbf{p}$ with the model distribution $\mathbf{p}_{\boldsymbol{\theta}}$. The perplexity (2.26) can also be interpreted as the (geometric) average branching factor of the language according to a language model (Rosenfeld, 2000), or of the target language given the aligned source sentence in the case of a translation model. The use of perplexity has an advantage as it is easy to compute and to optimize. The perplexity optimization leads straightforwardly to the conventional maximum log-likelihood procedure (Section 3.1). Moreover, when multiple LMs or TMs exist, they can be interpolated according to the coefficients which are trained by the EM algorithm to minimize the perplexity on a validation set (see, for example (Schwenk et al., 2007)).

In spite of all these advantages, the quality of a language or translation model must be measured ultimately by its impact on the application for which it has been designed. However, the correlation between perplexity and the performance measures is not obvious in many cases. For ASR systems, Rosenfeld (2000) claim that a reduction of 5% in perplexity is usually not practically significant; a $10-20\%$ reduction is noteworthy, while only a perplexity improvement of 30% or more is significant and can be translated into performance improvements. Perplexity has been shown in (Chen et al., 1998) to predict well WER when considering only n-gram models trained on in-domain data. The authors propose a new metric by imitating the WER calculation without the use of a speech recognizer, and show that this metric is superior when comparing LMs from different categories. Although the perplexity continues to be widely used, some recent attempts have been made to train models directly based on the application quality metric (He and Deng, 2012; Auli and Gao, 2014; Do et al., 2014a).

2.5.2 Extrinsic evaluation and integration of CSMs

Extrinsic evaluation consists of incorporating the CSM into a SMT system as an additional feature function, and using the final system performance, expressed for instance in terms of BLEU score, to assess the quality of the model. This section describes several strategies that can be used to integrate a CSM into the translation system.

N-best and lattice rescoring

Because of a high computational cost during the inference, CSMs are often integrated into SMT systems in a two-step approach : first, a set of best translation candidates is generated for each source sentence; then, the CSM is used to rescore this set, resulting in a new score for each hypothesis. The added feature, along with other scores of the baseline system, are tuned on the development set (Section 1.3.1) in order to optimize the log-linear coefficients $\boldsymbol{\lambda}$ according to the BLEU score. The whole system is then used to process the test data. During this procedure, the set of best hypotheses can be stored in

N-best lists, or more compactly in lattices.

For each source sentence $\mathbf{s}$, we assume that the baseline SMT decoder generates a set of top translation candidates $\mathbf{h}_1, ..., \mathbf{h}_N$. Each hypothesis $\mathbf{h}_i = (\mathbf{t}_i, \mathbf{a}_i)$ is associated with a score from Equation (1.3) :

$$F_{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{h}_i) = \sum_{m=1}^{M} \lambda_m f_m(\mathbf{s}, \mathbf{h}_i) \tag{2.27}$$

Suppose that the CSM with parameters $\boldsymbol{\theta}$ outputs a score $g_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)$ for each hypothesis $\mathbf{h}_i$ (which typically corresponds to the log-probability of the derivation, $g_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i) = \log \mathbf{p}_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)$). This score is added as $(M+1)^{th}$ feature function in Equation (2.27), giving a new scoring function :

$$G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i) = F_{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{h}_i) + \lambda_{M+1} g_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)$$

which depends both on the CSM parameters $\boldsymbol{\theta}$, as well as on the coefficients $\boldsymbol{\lambda}$ of the scoring function. Tuning $\boldsymbol{\lambda}$ can use standard tools presented in Section 1.3. Because of its simplicity, N-best list rescoring is widely used to incorporate new models to the SMT system (Schwenk et al., 2007; Kalchbrenner and Blunsom, 2013; Hu et al., 2014). Another advantage is that the procedure can score the whole derivation without any assumption regarding the decomposition of the translation process. However, a limitation of this approach is that N-best lists contain hypotheses that are typically very redundant with little diversity, representing only a few combinations of translation decisions (Huang, 2008). Moreover, in the discriminative training of the log-linear coefficients (Section 1.3), the list of hypotheses is often considered as a proxy for the whole space of derivations (Chiang, 2012), their limited variety may be harmful to the tuning step.

Compared to the N-best representation of hypotheses, lattice or hypergraph-based structures offer a much richer representation of hypotheses produced by the decoder, since they compactly encode an exponential number of hypotheses in polynomial space (Macherey et al., 2008). Auli et al. (2013) propose a new lattice rescoring algorithm to integrate a recurrent neural network joint model [17]which improves by 1.5 BLEU points on the WMT'12 French-English task. However, their experiments show no significant improvement of the lattice rescoring compared to the N-best list rescoring, while the former is performed at a higher computational cost.

Decoding with neural network models

A more plausible solution to work around the shallow representation of N-best lists is to use the CSM during the first decoding phase to build the search space. The first work in which a large-vocabulary neural network model is integrated into the decoding is (Vaswani et al., 2013). In this paper, the authors extend the NNLM of (Bengio et al., 2001, 2003a) in two ways : first, the softmax output layer is replaced by a *self-normalized* NCE output layer (Section 3.2); second, the non-linear function f uses rectified linear

[17].

units with cheaper computation than the sigmoid or the tanh functions. The CSM integration is then performed either by N-best list rescoring or by directly decoding with the NNLM in order to increase its influence on the SMT system. The second approach is shown experimentally to improve upon the first one by up to 1.0 BLEU point on a Chinese-English task, but the improvements are not significant on French-, German- and Spanish-to-English tasks. Later, Devlin et al. (2014) also include a feed-forward CSTM to the decoder, giving a strong improvement of 3.0 BLEU points over a baseline which has already incorporated a powerful NNLM. The improvement is attributed not only to the extension of the conditioning part over source context words, but also to the direct decoding with CSTM.

Work on direct decoding with CSMs needs to address the difficulty related to the computational complexity of the model inference. As scores need to be computed at run-time, most work use intensively techniques that have been described in Section 2.3.4 to speed up the computations. For the output layer, *un-normalized* and *self-normalized* large-vocabulary output layers are widely used. Later works, such as the implementation of CSTMs as a feature in MOSES (Birch et al., 2014) prefer using the NCE approach as it speeds up not only the inference but also the training. The resulting integrated decoder is reported to be twice slower than the one without CSM, which makes the whole system quite reasonable in terms of processing time. For the hidden layer, *context grouping* is not really useful as the CSM is often required to process only one word at a time. Instead, *caching* (Vaswani et al., 2013) and *pre-computation* (Devlin et al., 2014) can be used to reduce costly operations at run-time, and have been shown to be very efficient. However, the condition for these methods to work well is that the neural structure should be kept to the conventional feed-forward network with preferably only one hidden layer. It is also noticed that integrating the Restricted Boltzmann Machine (RBM)-based LM into decoding could be done only with an approximation of RBM probabilities which avoids the normalization (Niehues and Waibel, 2012). Finding the right trade-off between, on the one hand complex neural structures for modelling long-span dependencies, and on the other hand simpler feed-forward network without costly operation, is an interesting research line.

2.5.3 Discussions

It is important to notice that, in the ideal situation, the training procedure should be directly related to the final performance evaluation metric. Unfortunately, in spite of the emergence in the literature of different integration strategies, the dominant training method for CSTMs is still the maximization of the conditional likelihood which corresponds to the minimization of perplexity. This training objective function is simple, however only loosely correlates with the MT evaluation metrics (such as BLEU score) and makes the training step totally separated from its intended use. As a consequence, during the training phase, we have only few indications about the translation performance, apart the perplexity on held-out data. This loose correlation between the training and testing phases may raise doubts about the performance comparisons reported in the literature, as long as models are not trained for tasks they are assigned. For instance, while Devlin et al. (2014) conclude that decoding using CSTMs outperforms the N-best rescoring, it might be a consequence of the fact that the models have not been trained to

optimally perform in N-best rescoring. Moreover, if information about other components of the SMT system were given to the training phase, it is likely that they would perform even better during the decoding step.

In this dissertation, our main goal is to investigate new training strategies which take into account the impact of the CSM on the translation performance. As the N-best list rescoring is the simplest integration mode, we investigate CSMs mainly within this framework. Its simplicity allows us to deduce a correlated training procedure (Chapter 5) inspired from the discriminative SMT (Section 1.3), and to quantitatively assess the model performance at the least cost. It is interesting however for future work to further experiment with other integration modes, such as the direct decoding with SMT systems, or even the direct translation with neural network-based models.

2.6 Conclusions

This chapter gives a description of different neural network model structures that are used in the language and translation modelling tasks. The use of these continuous-space models has been motivated firstly by the estimation problem related to discrete language models (discrete LMs). Conventionally, traditional n-gram LMs rely on the maximum-likelihood estimate (MLE), resulting in an estimation procedure based on relative frequencies. However, such estimates are often not statistically reliable, as the number of free parameters increases exponentially with the degree n of the model. *Smoothing* has been the most popular technique used to correct these MLE probabilities by redistributing the probability mass from frequent word sequences to less frequent ones, through either *backoff* or *interpolation* techniques. However, such approaches still suffer from the fundamental limitation which lies in the estimation based on statistics of discrete events from a finite set.

The *data sparsity* problem raises the necessity of reducing the number of free parameters, for instance via a class-based approach which aims at modelling the relationship between discrete events with an event clustering. While the last approach has been shown to have some usefulness, continuous-space models (CSMs), via artificial neural networks, seem to offer a more systematic way of modelling such relationships, as they exploit continuous representations (embeddings), expressed naturally in weight matrices, and which can be learnt jointly with other model parameters. During the last few years, different neural structures have been proposed for the language modelling task, such as feed-forward or recurrent networks; each has its own way in handling context words and using the context in the word prediction. Even though this category of models has been proved to boost the performance of applications for which they are designed, CSMs are often associated with very expensive training and test procedures, mainly because of the normalization at the output layer, for which particularly intensive efforts have been made to workaround the computational problem (Section 2.3.4).

Besides the language modelling task, CSMs have also been used in translation models, which differ from the first task by the inclusion of context words from a foreign language. The application of CSMs in TMs also serves as a way to overcome the limitations of the maximum-likelihood estimate over word or phrasal units. Indeed, the *data sparsity*

problem in TMs is particularly severe, as training data is reduced, whereas the models often require to manipulate much larger vocabularies (such as word or phrase pairs) than the vocabulary of words.

Recent experimental results in the literature clearly suggest that it is preferable to factorize the phrase or phrase pair translation units into smaller units, such as words; the most successful CSTMs use words as minimal units. These *lexicalized* models can therefore be viewed as the bilingual generalization of NNLMs where vocabulary words come from two different languages. The conditioning on context source words is an important feature which distinguishes the CSTMs from each other, and from the target NNLMs. Among various factors which determine the performance of a CSTM, strategies for choosing source words which are included in the context seem to be of primary importance. Various model structures proposed for NNLMs can also be applied to CSTMs. However, in the scope of this dissertation, we base our experiments on the n-gram-based CSTM proposed in (Le et al., 2012a) which conditions the probability on context words determined by the boundaries of tuples (Section 2.4.2).

In spite of different proposals and ideas, there are two problems with CSTMs that still have not been resolved satisfactorily. The first one concerns the cost of training and inference, which hinders practical uses of CSTMs no matter how the models are incorporated in applications. The second problem consists in the gap between the training and testing phases. Although CSMs can be evaluated both by intrinsic (perplexity) and extrinsic measures, the last ones, based on the performance of the application for which CSMs have been designed, are more promising. While the correlation between these two measures is not obvious in many cases, in practice most CSTMs still employ the maximum-likelihood training method which is consistent with the perplexity, but not with the translation performance. It is hence likely that the models are not optimally optimized to perform well inside a SMT system along with other components.

3

Maximum Conditional Likelihood and Noise Contrastive Estimation

CSMs are effective in boosting the performance of NLP applications, however they also come along with expensive training and inference procedures. A major part of the computational cost comes from the handling of large vocabularies in the output layer (Section 2.3.4). While this problem can be avoided by using a smaller *short-list* containing only the most frequent words (Schwenk and Gauvain, 2004; Schwenk et al., 2007), this solution is not satisfactory when all vocabulary words need to be handled and evaluated by the model. Several other workarounds have been proposed in the literature, such as the ranking LM (Section 2.3.3), or the training procedure described in (Devlin et al., 2014) aimed at equipping the ranking LM with a probabilistic interpretation. However, these methods suffer from the lack of a probability estimation which limits their efficiency in inference, or from an expensive training.

In this chapter, we describe in details two main approaches that help to efficiently handle arbitrarily large vocabularies in neural network models. While vocabulary words can be hierarchically organized as in the SOUL structure, the second approach employs a sampling scenario to approximately estimate the gradient of the objective function used in *Stochastic Gradient Descent* (SGD). While both methods have been shown to be efficient in integrating large vocabularies without slowing the inference, there is still no systematic comparison between them, in terms of training and inference speed and their uses in applications. We will analyse the advantages and weaknesses of each method; the comparison is carried out on various training and integration situations. In each experiment, the intrinsic measure (perplexity) is first used to evaluate the models. However, the effectiveness of CSMs must finally be assessed through their impact on the applications for which they have been designed. Moreover, each of these approaches comes with a variety of choices of hyper-parameters which can have an impact on the performance of the final system.

The chapter begins with a description of the maximum-likelihood estimation (MLE) used to train the conventional CSM (Bengio et al., 2001, 2003a), and of the SOUL structure. The second section describes several methods for approximating the MLE gradient based on *Importance Sampling* algorithms, the most effective one being *Noise Contrastive Estimation* (or NCE). Finally, comparative experiments are performed (Section 3.3) on 3 different domains, which represent various situations of training and building systems in SMT. Part of these experimental results have been published in the description paper of the joint KIT-LIMSI submission to the WMT'2015 English-to-German shared translation task (Ha et al., 2015).

3.1 Maximum likelihood training and SOUL structure

Conventional neural network language models (NNLMs) are trained to maximize the conditional likelihood (CLL) on training samples (Bengio et al., 2003a; Schwenk and Gauvain, 2004). In this framework, the training corpus is considered as a set of n-grams (for n-gram language and translation models [1]), or more generally as a set of pairs composed of a context $\mathbf{c}$ and a word w to be predicted. Let us denote this set by $\mathcal{S}_n$, the likelihood computed based on the CSM free parameters $\boldsymbol{\theta}$ is the product of likelihoods estimated on each n-gram :

$$\mathbf{p}(\mathcal{S}_n|\boldsymbol{\theta}) = \prod_{(w,\mathbf{c})\in\mathcal{S}_n} \mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c})$$

hence, the maximization of the likelihood is transformed to the minimization of the following CLL criterion :

$$\mathcal{L}_{cll}(\boldsymbol{\theta}, \mathcal{S}_n) = \sum_{(w,\mathbf{c})\in\mathcal{S}_n} -\log \mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c}) + \mathcal{R}(\boldsymbol{\theta}) \tag{3.1}$$

where $\mathcal{R}(\boldsymbol{\theta})$ is the $\mathcal{L}_2$-regularization term defined by :

$$\mathcal{R}(\boldsymbol{\theta}) = \gamma \times \frac{||\boldsymbol{\theta}||^2}{2} \tag{3.2}$$

and γ is an associated hyper-parameter.

This criterion corresponds to the negated conditional log-likelihoods of all n-grams in $\mathcal{S}_n$. The most costly phase in the optimization of this criterion is the computation of conditional probabilities which requires to normalize the network activations of all possible output words in $\mathcal{V}^o$ with the following *softmax* function :

$$\mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c}) = \frac{\mathbf{e}_{\boldsymbol{\theta}}(w,\mathbf{c})}{\sum_{w'\in\mathcal{V}^o} \mathbf{e}_{\boldsymbol{\theta}}(w',\mathbf{c})} = \frac{e^{\mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})}}{\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c})} \tag{3.3}$$

where, $\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c}) := \sum_{w'\in\mathcal{V}^o} e^{\mathbf{a}_{\boldsymbol{\theta}}(w',\mathbf{c})}$ and $\mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})$ denotes the activation of the neural network corresponding to the word w and computed with the input from context $\mathbf{c}$. This normalization intervenes both in the training and inference of the CSM. The operation is

[1] In the scope of this dissertation, we always consider n-gram models in our experiments.

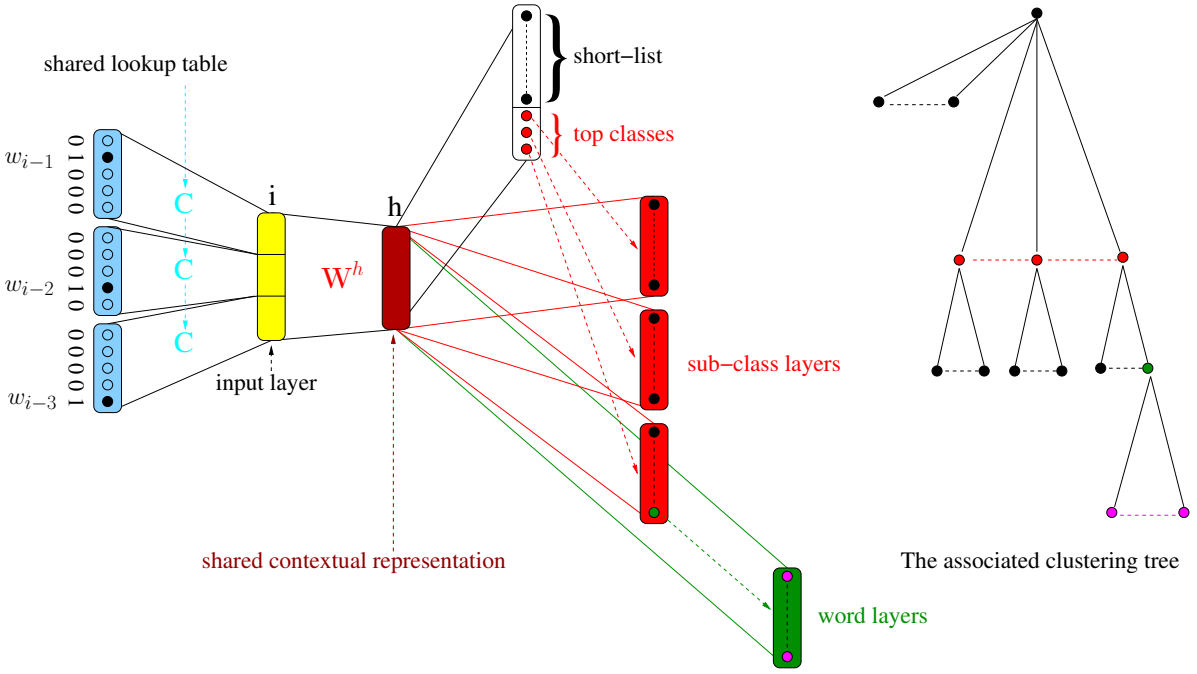


Figure 3.1 – SOUL structure via the organization of vocabulary words into clusters, as implemented and described in (Le et al., 2011, 2013).

extremely costly for large output vocabulary. Indeed, the number of floating points operations (according to (Schwenk and Gauvain, 2004)) needed to compute such a probability is :

$$((n-1) \times \mathrm{D} + 1) \times \mathrm{H} + (\mathrm{H} + 1) \times |\mathcal{V}^o| \tag{3.4}$$

where we reuse the notations that have been described in Section 2.3.1 : D is the dimension of word embeddings, H denotes the size of the last hidden layer, and n is the degree of the model. As in Automatic Speech Recognition and Machine Translation, the size of $\mathcal{V}^o$ is often much larger than D and H, the quantity can be asymptotically approximated with :

$$\mathrm{H} \times |\mathcal{V}^o| \tag{3.5}$$

which is also the asymptotic complexity of processing the output layer during training, and is proportional to the output vocabulary size.

3.1.1 Description of Structured OUtput Layer

As the output layer represents the major part in the computational cost (3.4), a modification of this layer is necessary to efficiently estimate word probabilities. Early work along these lines (Morin and Bengio, 2005; Mnih and Hinton, 2008) proposes to organize words in $\mathcal{V}^o$ hierarchically : each word is attached to a leaf in a binary tree structure, hence predicting output words is equivalent to the prediction of paths that lead from the root to leaves. The idea is inspired from the class-based LM (Brown et al., 1992; Kneser and Ney, 1993; Rosenfeld, 2000) applied to continuous-space models.

The SOUL structure (Le et al., 2011, 2013) is a combination of these two ideas. In general, each word w in $\mathcal{V}^o$ is assigned to a cluster Cl_w, then the word probability is

Algorithm 3 SOUL training scheme described in (Le, 2012).

1: **Step 1, short-list pre-training** provides a first estimate of word representation by training an NNLM covering a short-list.
2: **Step 2, word clustering** derives a clustering from the information in the word representation obtained from Step 1.
3: **Step 3, out-of-short-list (OOS) pre-training** trains an NNLM with OOS words as output.
4: **Step 4, full training** trains a SOUL NNLM with the clustering derived from Step 2, and with the parameters initialized using the parameters obtained at Step 3 and Step 1.

estimated first by estimating the probability of this cluster :

$$\mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c}) \underset{\text{SOUL}}{:=} \mathbf{p}_{\boldsymbol{\theta}}(\text{Cl}_w|\mathbf{c}) \times \mathbf{p}_{\boldsymbol{\theta}}(w|\text{Cl}_w, \mathbf{c}) \tag{3.6}$$

Note that words in Cl_w can in turn be organized in clusters; this class-based approach therefore represents the output vocabulary $\mathcal{V}^o$ as a clustering tree. The word prediction problem is hence translated into sub problems of predicting a sequence of decisions that form a path from the root to a leaf. In particular, these sub-problems share as much as possible their free parameters so that the total number of parameters is limited. More precisely, the two probabilities are both conditioned on $\mathbf{c}$, hence both use the vector representation of the context as their input (see Figure 3.1); the standard network structure up to the last hidden layer is shared among all predictions. The conditioning on Cl_w of the second probability in (3.6) is materialized by the creation, for each cluster, of a separate output layer with its own weight matrix and bias vector. The relationship between clusters can be expressed in terms of distances between these continuous representations. Figure 3.1 gives an illustration of the SOUL structure used in practice.

Let n_c be the number of word clusters, and $\underset{w\in\mathcal{V}^o}{\text{argmax}}\,|\text{Cl}_w|$ the maximal number of words contained in each cluster, then the asymptotic cost (3.5) becomes :

$$\text{H} \times \text{n}_c + \text{H} \times \underset{w\in\mathcal{V}^o}{\text{argmax}}\,|\text{Cl}_w| \tag{3.7}$$

A well-chosen clustering scenario can thus reduce drastically the asymptotic cost. For instance, if all clusters contain approximatively the same number of words, then we have $\text{n}_c \times \underset{w\in\mathcal{V}^o}{\text{argmax}}\,|\text{Cl}_w| \approx |\mathcal{V}^o|$. Moreover, if the number of clusters is chosen to be close to the number of words in each cluster, then $\text{n}_c \approx \underset{w\in\mathcal{V}^o}{\text{argmax}}\,|\text{Cl}_w| \approx |\mathcal{V}^o|^{1/2}$, hence the asymptotic cost becomes $2\text{H}\times|\mathcal{V}^o|^{1/2}$, instead of $\text{H}\times|\mathcal{V}^o|$. With a 500K-word vocabulary, the speed-up is about 350x faster than the standard structure with only one softmax output layer.

3.1.2 Initializing SOUL models

The speed-up provided by the SOUL structure comes along with the additional cost of building the hierarchical structure over $\mathcal{V}^o$. The proposal in (Morin and Bengio, 2005) builds this structure using an expert knowledge (WordNet). Later, the log-bilinear LM

described in (Mnih and Hinton, 2008) and the SOUL structure in (Le et al., 2011) both learn this hierarchy via the word embeddings. More precisely, a NNLM with SOUL structure can be built through the 4-step procedure described in Algorithm 3. The first step consists of training an NNLM with a short-list output (but still with full input) in order to obtain an embedding for each word in the vocabulary. These vectors are then used to run a clustering algorithm (typically K-mean) which delivers the structure that will accelerate the training in steps 3 and 4.

Initializing SOUL CSTM :

The SOUL CSTM, described in Section 2.4.2, is initialized from two SOUL language models, one in the source and another in the target language. Here we only consider the lexicalized models involved in the third formulation (2.23) of Section 2.4.2. Depending on predicted words and contexts, the bilingual model can be considered as an extension of the monolingual model on the source or target language. For instance, the model estimating $\mathbf{p}(t_l^k|\mathbf{c}_{n-1}(t_l^k), \mathbf{c}_{n-1}(s_{l+1}^1))$ is initialized from a target SOUL NNLM. Its output layers are copied from the target NNLM, as both models predict the same set of words; while the look-up table and hidden layers are extended to also include vocabulary words from the source language, using parameters of the source SOUL NNLM.

3.2 Noise contrastive estimation

In this section, we consider another form of training called *Noise Contrastive Estimation*, or NCE. This algorithm belongs to the class of sampling-based training methods which are useful to learn exponential models over a large set of elements (Gutmann and Hyvärinen, 2010). Mnih and Teh (2012) propose to apply this method to NNLMs.

3.2.1 Sampling-based methods for the training of NNLMs

The general idea of sampling-based methods is derived from the gradient computation of the conditional log-likelihood (CLL) criterion (3.1). Indeed, with word probabilities normalized by (3.3), the gradient of the log-probability is computed as follows :

$$\frac{\partial \log \mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c})}{\partial \boldsymbol{\theta}} = \sum_{w' \in \mathcal{V}^o} (\mathbb{I}_{w'=w} - \mathbf{p}_{\boldsymbol{\theta}}(w'|\mathbf{c})) \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w', \mathbf{c})}{\partial \boldsymbol{\theta}} \tag{3.8}$$

The use of Stochastic Gradient Descent (SGD) means that we are increasing the CLL of w and decreasing the CLL of all other words $w' \neq w$. The difficulty of optimizing a CLL criterion lies in the following fact :

(a) the explicit computation of $\mathbf{p}_{\boldsymbol{\theta}}(w'|\mathbf{c})$ requires the computation of the normalization constant, and

(b) the whole vocabulary is involved in the computation of the sum.

On the other hand, sampling-based methods aim to overcome (b) by penalizing the scores only of *a few negative examples*. These negative words are sampled from a probabilistic distribution which is simpler than the one that needs to be estimated, and from which a tractable sampling routine can be derived. By careful arrangements, some of these methods also allow us to ignore (a) by only requiring the activations corresponding to sampled negative examples, without knowledge of the normalization constant. In general, a sampling-based method is specified by the definition of :

(1) a sampling routine for negative examples, and

(2) an approximation of the CLL gradient that *rewards* the positive word w and *penalizes* negative ones. This can be seen as an approximation of the partition function over a reduced set of negative examples.

3.2.2 Importance Sampling

In order to estimate the CLL gradient (3.8), *Importance Sampling* (Robert and Casella, 2013) can be employed. The procedure comes from the rewriting of (3.8) as :

$$\begin{aligned}\frac{\partial \log \mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c})}{\partial \boldsymbol{\theta}} &= \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})}{\partial \boldsymbol{\theta}} - \sum_{w' \in \mathcal{V}^o} \mathbf{p}_{\boldsymbol{\theta}}(w'|\mathbf{c}) \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w',\mathbf{c})}{\partial \boldsymbol{\theta}} \\ &= \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})}{\partial \boldsymbol{\theta}} - \mathbb{E}_{w' \sim \mathbf{p}_{\boldsymbol{\theta}}(.|\mathbf{c})} \left(\frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w',\mathbf{c})}{\partial \boldsymbol{\theta}} \right)\end{aligned}$$

The expensive sum, which is also an expectation, can be estimated by Monte Carlo sampling methods (Robert and Casella, 2013) which consists of sampling examples from $\mathbf{p}_{\boldsymbol{\theta}}(.|\mathbf{c})$ and taking the average of their gradients. However, the underlying distribution $\mathbf{p}_{\boldsymbol{\theta}}(.|\mathbf{c})$ here is being estimated, hence we sample from another distribution $Q(.)$:

$$\begin{aligned}\frac{\partial \log \mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c})}{\partial \boldsymbol{\theta}} &= \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})}{\partial \boldsymbol{\theta}} - \sum_{w' \in \mathcal{V}^o} Q(w') \times \frac{\mathbf{p}_{\boldsymbol{\theta}}(w'|\mathbf{c})}{Q(w')} \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w',\mathbf{c})}{\partial \boldsymbol{\theta}} \\ &= \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})}{\partial \boldsymbol{\theta}} - \mathbb{E}_{w' \sim Q(.)} \left(\frac{\mathbf{p}_{\boldsymbol{\theta}}(w'|\mathbf{c})}{Q(w')} \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w',\mathbf{c})}{\partial \boldsymbol{\theta}} \right)\end{aligned}$$

where the last expectation term is estimated by sampling m examples $\hat{w}_i, 1 \leq i \leq m$ from $Q(.)$:

$$\frac{\partial \log \mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c})}{\partial \boldsymbol{\theta}} \approx \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})}{\partial \boldsymbol{\theta}} - \frac{1}{m} \sum_{i=1}^{m} \frac{\mathbf{p}_{\boldsymbol{\theta}}(\hat{w}_i|\mathbf{c})}{Q(\hat{w}_i)} \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(\hat{w}_i,\mathbf{c})}{\partial \boldsymbol{\theta}} \tag{3.9}$$

3.2.3 Biased Importance Sampling

Unfortunately, the formulation (3.9) still does not solve the problem (a) as we still have to compute explicitly the value of $\mathbf{p}_{\boldsymbol{\theta}}(\hat{w}_i|\mathbf{c})$. Instead, the sum can be estimated by a biased

approximation which is based on the particular form of probabilities computed by the standard CSM (Equation (3.3)) :

$$\mathbb{E}_{w' \sim Q(.)} \left(\frac{\mathbf{p}_{\boldsymbol{\theta}}(w'|\mathbf{c})}{Q(w')} \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w', \mathbf{c})}{\partial \boldsymbol{\theta}} \right) = \mathbb{E}_{w' \sim Q(.)} \left(\frac{1}{\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c})} \times \frac{\exp(\mathbf{a}_{\boldsymbol{\theta}}(w', \mathbf{c}))}{Q(w')} \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w', \mathbf{c})}{\partial \boldsymbol{\theta}} \right)$$
$$\approx \frac{1}{\mathbf{T}} \sum_{i=1}^{m} T(\hat{w}_i) \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(\hat{w}_i, \mathbf{c})}{\partial \boldsymbol{\theta}}$$

where $T(\hat{w}_i) = \frac{\exp(\mathbf{a}_{\boldsymbol{\theta}}(\hat{w}_i), \mathbf{c})}{Q(\hat{w}_i)}$, and $\mathbf{T} = \sum_{i=1}^{m} T(\hat{w}_i)$. This approximation is called *Biased Importance Sampling*, and is used in situations where the considered distribution (here, $\mathbf{p}_{\boldsymbol{\theta}}(.|\mathbf{c})$) can be computed up to a multiplicative constant (Kong et al., 1994). It is important to note that the estimator is biased, but its bias decreases when m increases, and can be shown to converge to the true expectation. For each sampled example $\hat{w}_i$, we do not need to compute $\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c})$, but only the activation $\mathbf{a}_{\boldsymbol{\theta}}(\hat{w}_i, \mathbf{c})$. The sampling distribution $Q(.)$ is chosen so that it is cheap to sample from, such as the unigram distribution (Bengio and Senécal, 2008).

Bengio et al. (2003b) and Bengio and Senécal (2008) propose to use Biased Importance Sampling for training NNLMs. However, the procedure only computes biased estimates which limits severely its use in practice. On the one hand, the bias decreases and approaches zero when m tends to infinity. On the other hand, we are not free to take an arbitrarily large m in order to benefit from this property. Indeed, m cannot be greater than the number of vocabulary words, otherwise the use of a sampling-based method is meaningless. Moreover, the corresponding variance is not guaranteed to be bounded (Ortiz and Kaelbling, 2000), meaning that the updates may be very unstable. As a consequence, the required number m of samples needs to be quite large, or adapted during the training. Bengio et al. (2003b) propose a method to estimate m that is necessary for the Biased Importance Sampling to yield the same variance as the standard version : if this number is greater than $|\mathcal{V}^o|$, then a switching back to full back-propagation is needed. Their experiments show that when fixing $Q(.)$ to the unigram distribution, this number increases exponentially as training progresses. Further improvement in (Bengio and Senécal, 2008) consists of adapting $Q(.)$ so that it stays close to the data distribution estimated by the neural network model : with the new adaptive scenario, the necessary number of samples grows only linearly, instead of exponentially. Compared to the full vocabulary back-propagation , the technique presented in (Bengio and Senécal, 2008) shows speed-ups up to 100x faster, when both m and $Q(.)$ are adapted during the training.

3.2.4 Noise contrastive estimation

Another approach to overcome the problem (a) in the formulation (3.9) is to set $Q(.)$ to a mixture of $\mathbf{p}_{\boldsymbol{\theta}}(.|\mathbf{c})$ and a *noise distribution* $\mathbf{p}_N(.)$, from which the number of examples sampled from $\mathbf{p}_N(.)$ is supposed to be K times greater than from $\mathbf{p}_{\boldsymbol{\theta}}(.|\mathbf{c})$:

$$Q(\hat{w}_i) \underset{\text{NCE}}{:=} \frac{\mathbf{p}_{\boldsymbol{\theta}}(\hat{w}_i|\mathbf{c}) + K\mathbf{p}_N(\hat{w}_i)}{K+1} \tag{3.10}$$

At first glance, sampling from (3.10) seems intractable as it involves $\mathbf{p}_{\boldsymbol{\theta}}(.|\mathbf{c})$. However, this distribution is being trained to approximate the data distribution, hence we can directly use the positive word w as an example from $\mathbf{p}_{\boldsymbol{\theta}}(.|\mathbf{c})$. The sampling routine is henceforth as easy as sampling from the *noise distribution*. By setting $m = K+1$ in (3.9) and using K examples $\{\hat{w}_i, 1 \leq i \leq K\}$ sampled from $\mathbf{p}_N(.)$, we have the following estimation of the CLL gradient :

$$\begin{aligned}
&\frac{\partial \log \mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c})}{\partial \boldsymbol{\theta}} \approx \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})}{\partial \boldsymbol{\theta}} - \\
&\frac{1}{K+1}\left(\frac{(K+1)\mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c})}{\mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c}) + K\mathbf{p}_N(w)} \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})}{\partial \boldsymbol{\theta}} + \sum_{i=1}^{K} \frac{(K+1)\mathbf{p}_{\boldsymbol{\theta}}(\hat{w}_i|\mathbf{c})}{\mathbf{p}_{\boldsymbol{\theta}}(\hat{w}_i|\mathbf{c}) + K\mathbf{p}_N(\hat{w}_i)} \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(\hat{w}_i,\mathbf{c})}{\partial \boldsymbol{\theta}}\right) \\
&= \frac{K\mathbf{p}_N(w)}{\mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c}) + K\mathbf{p}_N(w)} \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})}{\partial \boldsymbol{\theta}} - \sum_{i=1}^{K} \frac{\mathbf{p}_{\boldsymbol{\theta}}(\hat{w}_i|\mathbf{c})}{\mathbf{p}_{\boldsymbol{\theta}}(\hat{w}_i|\mathbf{c}) + K\mathbf{p}_N(\hat{w}_i)} \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(\hat{w}_i,\mathbf{c})}{\partial \boldsymbol{\theta}}
\end{aligned} \tag{3.11}$$

The optimization using the above approximative form is called *Noise Contrastive Estimation* (or NCE) (Gutmann and Hyvärinen, 2010), and has been first applied to the training of NNLMs in (Mnih and Teh, 2012). Like in the case of the Biased Importance Sampling, a large value of K reduces the estimation variance, hence makes the training more stable. But unlike the biased algorithm, the estimator corresponding to the NCE algorithm is always *unbiased* no matter the choice of $\mathbf{p}_N(.)$ and K. This feature of NCE, which derives from the standard Importance Sampling, constitutes its main advantage compared to the Biased Importance Sampling described in (Bengio et al., 2003b; Bengio and Senécal, 2008). While the choice of a noise distribution is still important, its adaptation, along with the adaptation of K during the training is no longer necessary to ensure convergence.

Self-normalization :

The approximate form (3.11) still contains the expensive probability terms $\mathbf{p}_{\boldsymbol{\theta}}(.|\mathbf{c})$. To speed-up the training phase, the work of (Mnih and Teh, 2012) proposes to replace $\mathbf{p}_{\boldsymbol{\theta}}(.|\mathbf{c})$ by the exponential of activations $\mathbf{e}_{\boldsymbol{\theta}}(.,\mathbf{c})$, i.e to set the partition function $\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c})$ to 1. Concretely, the *proposal* distribution of (3.10) becomes :

$$Q(\hat{w}_i) = \frac{\exp(\mathbf{a}_{\boldsymbol{\theta}}(\hat{w}_i,\mathbf{c})) + K\mathbf{p}_N(\hat{w}_i)}{K+1} \tag{3.12}$$

and the gradient is rewritten as follows :

$$\begin{aligned}
&\frac{\partial \log \mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c})}{\partial \boldsymbol{\theta}} \approx \\
&\frac{K\mathbf{p}_N(w)}{\mathbf{e}_{\boldsymbol{\theta}}(w,\mathbf{c}) + K\mathbf{p}_N(w)} \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c})}{\partial \boldsymbol{\theta}} - \sum_{i=1}^{K} \frac{\mathbf{e}_{\boldsymbol{\theta}}(\hat{w}_i,\mathbf{c})}{\mathbf{e}_{\boldsymbol{\theta}}(\hat{w}_i,\mathbf{c}) + K\mathbf{p}_N(\hat{w}_i)} \times \frac{\partial \mathbf{a}_{\boldsymbol{\theta}}(\hat{w}_i,\mathbf{c})}{\partial \boldsymbol{\theta}}
\end{aligned} \tag{3.13}$$

By forcing $\mathbf{e}_{\boldsymbol{\theta}}(.,\mathbf{c})$ to follow the data distribution in (3.12), a side effect is that the normalization constants tend to 1 as training progresses. At the end of the training, this *self-normalization* allows us to completely ignore the expensive normalization when computing model scores, hence speeding up considerably the inference.

This phenomenon is explained in (Mnih and Teh, 2012) by the fact that CSMs have a lot of free parameters that constraining the un-normalized output scores $\mathbf{e}_{\boldsymbol{\theta}}(., \mathbf{c})$ to estimate a distribution is easy. In order to better understand it, we reconsider the gradient (3.13) as deriving from the optimization of the following loss function [2] :

$$\mathcal{L}_{nce}(\boldsymbol{\theta}, (w, \mathbf{c})) = -\log \frac{\mathbf{e}_{\boldsymbol{\theta}}(w, \mathbf{c})}{\mathbf{e}_{\boldsymbol{\theta}}(w, \mathbf{c}) + K\mathbf{p}_N(w)} - \sum_{i=1}^{K} \log \frac{K\mathbf{p}_N(\hat{w}_i)}{\mathbf{e}_{\boldsymbol{\theta}}(\hat{w}_i) + K\mathbf{p}_N(\hat{w}_i)} \tag{3.14}$$

which has been obtained in (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012) from the CLL optimization transformed into a binary classification problem of distinguishing positive examples from negative ones sampled using the noise distribution. Suppose that we can choose a noise distribution that is different from $\mathbf{e}_{\boldsymbol{\theta}}(., \mathbf{c})$ only by a multiplicative constant : $\mathbf{p}_N(.) = \mathbf{e}_{\boldsymbol{\theta}}(., \mathbf{c}).H^{-1}$. Then, the loss (3.14) becomes :

$$\begin{aligned}\mathcal{L}_{nce}(\boldsymbol{\theta}, (w, \mathbf{c})) &= -\log \frac{\mathbf{e}_{\boldsymbol{\theta}}(w, \mathbf{c})}{\mathbf{e}_{\boldsymbol{\theta}}(w, \mathbf{c}) + K\mathbf{e}_{\boldsymbol{\theta}}(w, \mathbf{c})H^{-1}} - \sum_{i=1}^{K} \log \frac{K\mathbf{e}_{\boldsymbol{\theta}}(\hat{w}_i, \mathbf{c})H^{-1}}{\mathbf{e}_{\boldsymbol{\theta}}(\hat{w}_i) + K\mathbf{e}_{\boldsymbol{\theta}}(\hat{w}_i, \mathbf{c})H^{-1}} \\ &= (K+1)\log(H+K) - \log H - K\log K\end{aligned}$$

that has a gradient $\frac{K(H-1)}{H(H+K)}$ with respect to H, which is zero only when $H = 1$. It means that the corresponding loss function is minimized only when $\mathbf{e}_{\boldsymbol{\theta}}(., \mathbf{c})$ has the same normalization constant as the noise distribution. In practice, the sampling distribution derived from $\mathbf{e}_{\boldsymbol{\theta}}(., \mathbf{c})$ is unknown, but the minimization of (3.14) tends to close the gap between the partition functions of the two distributions. *Self-normalization* is hence achieved by imposing an exact form of the noise distribution in the mixed proposal (3.10).

3.2.5 Choosing a noise distribution

It is well-known that *Importance Sampling* has a reduced variance if the *proposal* distribution is close to the target distribution (Robert and Casella, 2013). In (Bengio and Senécal, 2008), more complex back-off bigram and trigram models are used in the biased version, but produce even poorer results than the simple unigram proposal. This may be because back-off n-gram models are very different from the corresponding models estimated by CSMs (Goodman, 2001). An adaptive scenario for $Q(.)$ has then been proposed in (Bengio and Senécal, 2008), but even in this case, the procedure requires sampling hundreds of samples for each training example, and this number increases rapidly at training progresses.

Being an *unbiased* estimator, the NCE algorithm also benefits from the mixture (3.10) where $Q(.)$ already incorporates the distribution estimated by the CSM, hence is made closer to the latter. Choosing the noise distribution is still important, but it can be decided once and for all (along with the number of negative examples) before training starts and kept fixed during all the training process without causing divergence. Experiments in (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012) show that with the use of a simple unigram distribution and only about 25 negative samples, NCE training can result

[2] This loss corresponds to one n-gram $(w, \mathbf{c})$. For $\mathcal{S}_n$, the objective function will be the sum of losses computed over all n-gram $\in \mathcal{S}_n$, along with a regularization term of (3.2).

in a model which is as good as the one trained by optimizing the CLL criterion. Even in the extreme case where uniform distribution is used with $K = 1$, the training does not diverge, which proves the NCE's advantage of being unbiased.

3.3 Comparison between SOUL and NCE

In this section, we will report experiments used to quantitatively compare the two approaches mentioned in the previous sections : the CLL optimization using SOUL structure, and the NCE algorithm. The first approach represents a model-specific solution for the computational problem of normalizing over a large vocabulary, while the second comes from the category of sampling-based approaches (Section 3.2.1). The models trained by these two methods will be called respectively SOUL and NCE models for simplicity. The performance of each method is evaluated by their perplexities [3] (Section 2.5.1) computed on validation sets, the amount of time needed to finish one training iteration, as well as the number of iterations until convergence. CSMs are also then incorporated into SMT systems using N-best rescoring (Section 2.5.2) to help improving the translation performance.

Our SMT experiments are carried out in three different domains. The first task is derived from the text translation track of IWSLT'2011 from English to French (the TED Talks task (Federico et al., 2012)). The baseline system has been trained in the condition of WMT'2013 shared translation task [4], whereas the training of NNLMs and CSTMs only uses an in-domain dataset containing 107,058 aligned sentence pairs. The second domain is the medical translation task of WMT'2014 [5] (English to French). For this task, we use all in-domain authorized corpora (above 5M sentence pairs) to build a MOSES-based [6] SMT system, while CSTMs are trained on 200K parallel sentence pairs extracted from the Patent-Abstract corpus. Finally, the third task represents a situation where multiple large corpora are available to train CSTMs: both the SMT system and CSTMs are built in the condition of the WMT'2015 shared translation task. [7] This experiment is extracted from the joined submission of KIT (Karlsruhe Institute of Technology) and LIMSI to the English to German translation task. Details on the baseline system, as well as on the CSTM structures can be found in the description paper (Ha et al., 2015).

Table 3.1 presents detailed information about our experimental set-ups, while Table 3.2 collects the values of hyper-parameters of our neural network structure (Section 2.4.2). For the TED Talks task, we train both NNLMs and CSTMs for the SMT system, while in other domains, only bilingual CSTMs are trained and incorporated into SMT systems. The official development and test sets for each domain are used, except for the first task where, following (Le et al., 2012a), we swap these two sets : the tuning of log-linear coefficients is carried out on 1,664 sentences of the official test, while the final test on 934 sentences of the original development set. Baseline SMT systems are built based

[3] Like in previous work on NCE NNLMs (Mnih and Teh, 2012; Vaswani et al., 2013), the computation of perplexities requires us to explicitly normalize over $\mathcal{V}^o$.

[4] http://www.statmt.org/wmt13/.

[5] http://www.statmt.org/wmt14/medical-task/.

[6] http://www.statmt.org/moses/.

[7] http://www.statmt.org/wmt15/.

Task	dev/test	SMT sys.	Init. CSM	Train data CSM	Voc. (src/trg)
TED NNLM	IWSLT-tst2010 / IWSLT-dev2010	WMT'13 n-code (12M)	random	107,058 of TED	506K En / 493K Fr
TED CSTM			from NNLMs		
Medical WMT'14	devel/test	MOSES on Medical (5M)	random	200K of Patent-Abstract	587K En / 367K Fr
WMT'15 En-De	News2013 / News2014	MOSES-based WMT	from NNLMs	all WMT corpora	432K En / 500K De

Table 3.1 – Details about experimental configurations used to assess the performance of SOUL and NCE models.

Hyper-parameter	Value
n	10
D	500
H	2 hidden layers with 1000 and 500 units

Table 3.2 – Hyper-parameters for CSTMs.

either on n-code [8] (an open implementation of the n-gram approach for SMT described in Section 1.1.3), or on a conventional phrase-based approach. [9] NNLM parameters are initialized randomly; the clustering of $\mathcal{V}^o$ into groups is learnt from the pre-trained word embeddings following the procedure described in Algorithm 3. CSTMs are initialized from NNLMs as described in Section 3.1.2, except the medical task where all bilingual models are initialized randomly. [10] All experiments cover large vocabularies (about 500K) extracted from the parallel training data used to train the baseline SMT systems. With such vocabularies, training the standard CSM structure with a full softmax output layer is prohibitive; SOUL or NCE models are both plausible alternative solutions as they make the training feasible in a reasonable time.

The adaptation of learning rates is important for the convergence speed as well as final perplexities. Chapter 4 will investigate this aspect in details. For simplicity, in this chapter we only use by default *Down Scheduling* for the training of SOUL models, and *Adjust Scheduling* for NCE models, as they are both very simple global learning rate adaptation scenarios which have shown good performance in our experiments. [11]

	SOUL	NCE			
		uniform		TED unigram	
		25	50	25	50
English LM, 505k vocabulary words	**108.4**	332.4	330.1	**108.4**	110.0
French LM, 492k vocabulary words	79.0	349.6	304.3	**78.5**	84.3

Table 3.3 – Perplexities on developments sets of SOUL models and of NCE models with different training configurations. Note that to compute the perplexity, NCE models are explicitly normalized.

Configurations of system				dev/IWSLT-tst2010	tst/IWSLT-dev2010
Baseline system (BS)				33.9	27.6
BS + SOUL LM				34.8	**28.7**
BS + NCE LM	uniform	25	norm	34.3	28.2
			unnorm	34.3	28.2
		50	norm	34.4	28.1
			unnorm	34.3	28.2
	TED unigram	25	norm	**34.9**	**28.7**
			unnorm	34.8	28.6
		50	norm	34.8	28.6
			unnorm	34.8	28.6

Table 3.4 – Results of using a continuous space language model to rescore N-best lists of the baseline system.

3.3.1 Experiments on TED Talks 2011 English → French

Monolingual models

Table 3.3 presents the perplexities obtained on the validation set (IWSLT-tst2010) with SOUL and NCE NNLMs trained on the English and French sides of TED Talks corpus. We investigate the impact of two NCE hyper-parameters : the noise distribution, and the number of negative words [12] sampled from this distribution (25 or 50 words). The results reflect the importance of this noise : while none of these configurations causes divergence, the unigram distribution obtains perplexities which are 70% less than those with the uniform distribution. Once the unigram noise is chosen, there is only slight difference between configurations with 25 and 50 negative examples. The best NCE models are equivalent to SOUL NNLMs in terms of perplexity.

The result of incorporating these models into the WMT translation system is presented

[8] perso.limsi.fr/Individu/jmcrego/bincoder .

[9] http://www.statmt.org/moses/

[10] In this configuration, SOUL models benefit from the fact that their word clusterings is not randomized but inherited from monolingual models.

[11] They are the two adaptive methods which obtain the best performance among global learning rate adaptation scenarios, as observed in the experiments of Chapter 4.

[12] According to descriptions in Sections 3.2.1 and 3.2.4, this number should correspond to $K + 1$. However, by *negative words*, here we refer only to the K words generated by the noise distribution $\mathbf{p}_N(.)$.

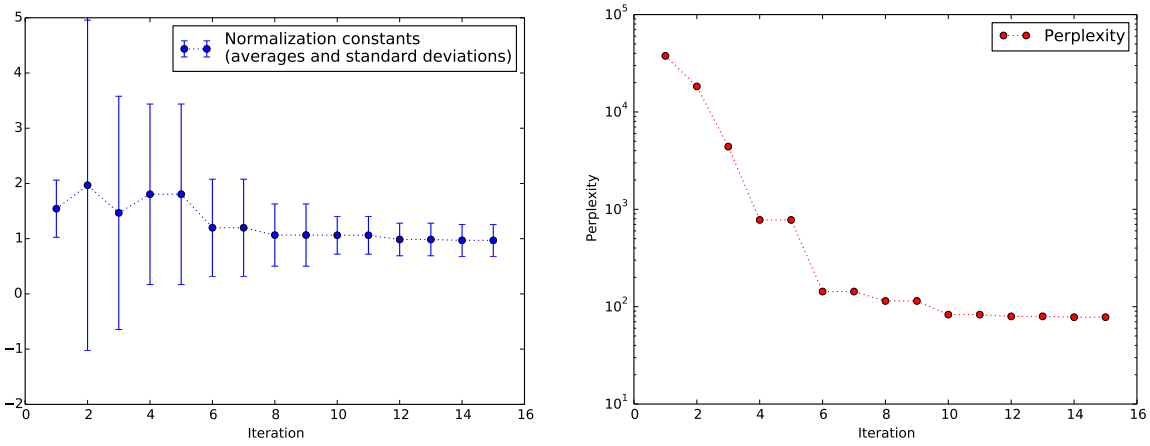
Training time (minutes) for each ite.			
	SOUL	NCE	
		25 neg.	50 neg.
	19.7	**19.1**	23.8
Time of decoding (minutes)			
Naive structure	SOUL	NCE	
275.6	7.8	**5.3**	

Table 3.5 – The time for each iteration (training) and for computing the scores of all hypotheses from N-best lists.

in Table 3.4. The French SOUL NNLM improves the baseline system by 0.9 BLEU points on the development, and by 1.1 BLEU points on the test set. These are equivalent to the performance obtained by NCE models which have been trained using the unigram noise distribution. Perplexity is known to be an intrinsic evaluation metric (Section 2.5.1) which is only partially correlated to the translation performance, especially when the latter also strongly depends on the baseline system. In our experiments, the 70% perplexity reduction observed in Table 3.3 translates only to a 0.5 BLEU point improvement. This comparison suggests that optimizing solely the perplexity will yield small improvements, and that the introduction of a new criterion which correlates better to the translation performance might be necessary to achieve more significant gains.

Interestingly, our comparison also includes the case where the scores of NCE models are explicitly normalized before being integrated to the SMT system; this case corresponds to the mention *norm* in the fourth column, while *unnorm* indicates that the activations are directly used as scores. Although taking up to more than 4 hours to complete the scoring on the development and test sets (the lower part of Table 3.5), *norm* does not give better results than *unnorm*, while the latter is 50x faster. This suggests that the un-normalized version of NCE models is not necessarily a weakness, and that the *self-normalizing* property of the NCE algorithm is practically sufficient for a successful integration into the SMT system. The computation of self-normalized scores is fast (even compared to SOUL models which normalize scores only on small softmax layers) and easy to be speeded up, using *pre-computation* techniques as described in (Devlin et al., 2014). This kind of models is henceforth promising if we want to integrate NNLMs during the runtime of the decoding, i.e to directly use these scores for the construction of the search space (Vaswani et al., 2013).

To further illustrate the self-normalization property of the NCE training, Figure 3.2a plots the averaged normalization constants (along with standard deviations) measured on the validation set after each training iteration. The figure clearly shows that the NCE algorithm, besides its approximation to the CLL gradient, also helps to make $\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c})$ close to 1 for each context $\mathbf{c}$, hence making the activations of the last layer ready for use in the translation system. Without this property, each normalization constant $\mathrm{H}_{\boldsymbol{\theta}}(\mathbf{c})$ would become a free parameter that needs to be optimized as described in preliminary experiments in (Mnih and Teh, 2012), or in (Devlin et al., 2014). This situation will slow severely the training, because the number of such parameters increases exponentially with the model order, n. It is important to note that, this self-normalization comes from the fact that we use a mixture of the CSM probability and the noise $\mathbf{p}_N(.)$ for the



(a) (b)

Figure 3.2 – (a) The evolution of normalization constants computed in all context word sequences from the validation set. These quantities are presented by their averages and standard deviations at each training iteration. (b) The evolution of perplexity measured on the validation set.

proposal distribution (3.10), and that the partition functions of these two distributions are "balanced" as a side effect of the NCE optimization.

Finally, Table 3.5 compares the time needed for SOUL and NCE models to finish one training epoch, as well as to perform inference on the test set. The training of NCE models becomes slower with 50 negative sampled examples, but $K = 25$ is sufficient to have good performance. For the inference, NCE models yield a computing speed which is $20 - 30\%$ faster than SOUL models, due to the fact that the scores of NCE models correspond to the activations directly computed by vector-vector (or matrix-vector with mini-batches) operations at the output layer, without any normalization. This modest speed-up can be explained by the fact that with the implementation of SOUL and NCE approaches, the computational cost of training and inference with the CSM is no longer dominated by the output layer, but by the processing of context words. Further improvements for NCE models would consist of *pre-computing* the contextual embeddings so that context words which have appeared in previous sentences can recover their vectorial representation without repeating entirely the forward step (Devlin et al., 2014).

The convergence speed is the main weakness of NCE, compared to the CLL optimization by SOUL structure. Due to the variance in estimating the CLL gradient by Importance Sampling, NCE training is known to require about twice as many epochs as the CLL optimization to converge (Mnih and Teh, 2012). Precisely, our SOUL NNLMs start to over-fit after 4 epochs, while the NCE training still have perplexities decreased after 15 epochs. However, as illustrated in Figure 3.2b, NCE models reach good perplexities around epoch 6. Even though they are not as good as those obtained by SOUL models, there is only slight difference between them which, as suggested by results in Table 3.4, will not translate into significant differences in the translation performance.

As a compensation, the NCE training is carried out on the standard NNLM structure which does not require any specific initialization scheme, while the initialization of SOUL models needs to perform the first estimate of the word clustering for SOUL structure.

	SOUL	NCE	
		uniform	TED unigram
TrgSrc	9.2	57.2	12.0
Src	84.4	152.3	105.0
SrcTrg	8.4	32.8	11.9
Trg	54.1	85.0	68.8

Table 3.6 – Perplexities of 4 neural n-gram translation models, trained by SOUL and NCE.

The convergence of NCE training can be further improved by adapting a learning rate for each block of parameters, as described in Chapter 4.

Bilingual models

Word-based bilingual models estimate the probability of a source or target word given a context sequence composed of both source and target words. These models are derived from Equation (2.23) in Section 2.4.2. The four probabilities correspond to four CSTMs in our experiments, which are denoted respectively by *TrgSrc*, *Src*, *SrcTrg* and *Trg*. Like NNLMs, CSTMs can be evaluated using the perplexity computed on a validation set : here the validation is carried out on sentences from the development set (IWSLT-tst2010), coupled with their references, and segmented into bilingual *tuples* (Section 2.4.2). Table 3.6 presents the perplexities obtained by SOUL and NCE bilingual models, whereas Table 3.7 shows the results of incorporating these models in the baseline SMT system. Similarly to what has been observed in the monolingual case, using the uniform noise distribution results in perplexities which are much larger than those obtained by the unigram distribution estimated on TED Talks corpus. This perplexity differences translate into BLEU improvements of about 0.6 points in the last case where all CSTMs are used in the translation system, but only of 0.3 points if we compare to the best result obtained by the uniform noise distribution (28.9). Consistent with previous observations, the NCE training yields perplexities which are $20-30\%$ higher than SOUL models, but this difference is not reflected in the translation performance. Finally, using normalized NCE model scores instead of un-normalized ones, and increasing the number of sampled negative words to 50 does not help to improve BLEU score obtained by the overall system.

3.3.2 Experiments on the medical task 2014 English → French

Experiments so far have been carried out on the small in-domain TED Talks corpus and have shown that the unigram noise distribution, estimated on the same corpus, gives good performance. In the second set of experiments, we further investigate the impact of noise distributions, estimated on different datasets, on the convergence of the NCE training. The data used to train the SMT system is the medical parallel corpora authorized for the medical translation task of WMT'2014, whereas a $200,000$-sentence subset of Patent-Abstract corpus is used to train the CSTMs. Three noise distributions are compared : the uniform, as well as two unigram distributions estimated respectively on all medical

	dev = IWSLT-tst2010	test = IWSLT-dev2010
	Baseline system	
	33.9	27.6
	Baseline + TrgSrc	
SOUL	35.1	**28.8**
NCE uniform	34.6	28.3
NCE TED unigram	34.7	28.7
NCE normalized TED unigram	34.8	28.7
NCE TED unigram 50	34.8	**28.8**
	Baseline + Trg	
SOUL	34.8	28.7
NCE uniform	34.6	**28.9**
NCE TED unigram	34.8	28.7
NCE normalized TED unigram	34.8	28.7
NCE TED unigram 50	35.1	28.8
	Baseline + TrgSrc + Src	
SOUL	35.3	28.7
NCE uniform	34.6	28.3
NCE TED unigram	35.0	28.9
NCE normalized TED unigram	35.2	**29.2**
NCE TED unigram 50	35.0	29.0
	Baseline + SrcTrg + Trg	
SOUL	35.2	**29.2**
NCE uniform	34.6	28.7
NCE TED unigram	35.1	28.9
NCE normalized TED unigram	35.2	28.9
NCE TED unigram 50	35.2	28.8
	Baseline + all NNTMs	
SOUL	35.6	29.1
NCE uniform	34.7	28.6
NCE TED unigram	35.4	**29.2**
NCE normalized TED unigram	35.4	**29.2**
NCE TED unigram 50	35.4	**29.2**

Table 3.7 – Results of integrating CSTM into the out-of-domain translation system. By default, NCE models are trained with 25 negative examples per word, and use un-normalized scores for rescoring, except in the fourth line of each configuration where scores are normalized, and in the fifth line where $K = 50$ is used.

	SOUL	NCE		
		uniform	all medical unigram	training medical unigram
TrgSrc	6.1	17.2	7.6	7.3
Src	53.5	188.3	xxx	64.2
SrcTrg	7.7	193.1	xxx	10.2
Trg	30.9	58.6	xxx	35.8

Table 3.8 – Perplexities on the validation set of models trained using SOUL and NCE, and by different configurations of NCE. Different noise distributions give strikingly different results in terms of perplexities. The case *xxx* means that the corresponding training causes divergence.

	dev = devel	test = test
	Baseline system	
	39.4	38.1
	Baseline + TrgSrc + Src	
SOUL	41.8	39.6
NCE uniform	40.8	39.1
NCE training medical unigram	41.9	**39.8**
	Baseline + SrcTrg + Trg	
SOUL	42.1	39.5
NCE uniform	41.1	39.1
NCE training medical unigram	41.7	**39.6**
	Baseline + all NNTMs	
SOUL	42.3	**39.9**
NCE uniform	41.3	39.4
NCE training medical unigram	42.4	39.6

Table 3.9 – BLEU scores in rescoring N-best lists from the baseline system.

corpora (*all medical unigram*), and on the 200,000 sentence pairs of Patent-Abstract corpus used to train the CSTMs (*training medical unigram*).

Table 3.8 presents the perplexities obtained with NCE models, compared with those obtained by SOUL models trained on the same corpus. As expected, the 4 SOUL models, which have been trained directly to optimize the CLL (hence, the perplexity) criterion, achieve the best perplexities on the validation set. Between the two unigram noise distributions, the one estimated on all medical corpora causes divergence for three (out of four) bilingual models. The only explanation is the inconsistency between the noise and data distributions : while the CSTMs are trained only on the small subset of medical corpora, these unigram probabilities are however estimated on all corpora and seem to be very different from the data distribution. This experiment shows that the choice of a noise distribution plays a crucial role in the NCE algorithm. Given such experimental evidence, it is important that the noise distribution fits the unigram distribution estimated on the training data.

Table 3.9 summarizes results obtained by integrating these CSMs into the SMT system using N-best rescoring. CSTMs can be incorporated to the SMT system by pair of models (*TrgSrc + Src* or *SrcTrg + Trg*) according to each of the two equations (2.23),

Algorithm 4 Procedure of each NCE training iteration.

1: Given dataset $\mathcal{D}$, an integer $K > 0$.
2: **Step 1, data resampling** samples a subset $\mathcal{S}_n = \{(w, \mathbf{c})\}$ from $\mathcal{D}$. This set contains n-grams, which are equivalent to pairs of a context $\mathbf{c}$ and a word w to be predicted from this context.
3: **Step 2, unigram estimation** estimates a unigram distribution $\mathbf{p}_N(.)$ based on the set of predicted words w in $\mathcal{S}_n$. This distribution will be used as noise distribution (Section 3.2.4).
4: **Step 3, negative word sampling** generates, for each $(w, \mathbf{c}) \in \mathcal{S}_n$, K negative words $\hat{w}_1, ..., \hat{w}_K$ from the noise distribution $\mathbf{p}_N(.)$. The set of n-grams $(w, \mathbf{c}) \in \mathcal{S}_n$, along with all generated samples form a new set $\mathcal{S}_n^{\text{NCE}}$ for the NCE training.
5: **Step 4, optimization** minimizes the objective function (3.14) on $\mathcal{S}_n^{\text{NCE}}$, along with the regularization term (3.2). The optimization method is SGD.

or all 4 models (the lowest part of the table). The upper part of the table shows the BLEU scores of the baseline system without any CSM. Integrating NCE CSTMs trained with a uniform noise improves the translation by $1 - 1.3$ BLEU points, even though the perplexities corresponding to these models are quite high. The NCE models trained using the unigram noise distribution (estimated on the training corpus) improves the baseline system by up to 1.7 BLEU points. These results are consistent to what has been obtained with SOUL CSTMs.

Because of the importance of estimating the unigram noise distribution on the training data which is used to train CSTMs, we sketch in Algorithm 4 the procedure which corresponds to one training iteration of the NCE algorithm. The procedure is applicable both for training NNLMs and CSTMs, and constitutes, by default, our version of NCE algorithm for the rest of this dissertation.

3.3.3 Experiments on WMT English → German

Experiments on the English-to-German shared translation task of WMT'2015 evaluation campaign represent a large-scale situation where both the SMT system and CSTMs are built on multiple large corpora. These corpora are extracted from Europarl Parliament (EPPS), News Commentary (NC) and the common part of web-crawled data (Common Crawl). This work was carried out in collaboration with researchers from Karlsruhe Institute of Technology (KIT). Details on the SMT system, as well as on the data processing can be found in (Ha et al., 2015). We train one model for each of the 4 bilingual model structures (*TrgSrc*, *Src*, *SrcTrg* or *Trg*), and for each corpus; the total number of models is 12. Each model is trained either using the SOUL structure, or the NCE algorithm (Algorithm 4). At each iteration, a subset of 10M n-grams is resampled from the corresponding corpus; all training processes are stopped after 15 iterations.

Table 3.10 compares SOUL and NCE models in terms of perplexity, whereas Table 3.11 presents the results of adding these models into the SMT system. Consistent with our previous experiments, the perplexities obtained with SOUL models are generally lower than those obtained with NCE models; however these differences do not translate into

Model	TrgSrc	Src	SrcTrg	Trg
On newsco corpus				
SOUL	11.9	96.3	11.2	75.0
NCE	16.6	108.5	13.8	93.4
On crawl corpus				
SOUL	11.8	120.1	12.7	107.7
NCE	20.3	133.5	16.5	139.9
On EPPS corpus				
SOUL	7.2	51.5	7.2	49.7
NCE	9.5	56.4	8.6	57.0

Table 3.10 – Perplexities of SOUL and NCE models trained on different corpora.

Training method	dev = News2013	test = News2014
	Baseline system	
	18.8	18.6
	Baseline + all 4 bilingual models	
SOUL	19.7	19.3
NCE	19.5	19.5
SOUL + NCE	19.5	19.5

Table 3.11 – BLEU scores in rescoring using SOUL and NCE models.

	training speed	inference speed
SOUL	1000 / s	15500 / s
NCE	1000 / s	19400 / s

Table 3.12 – Speeds of the training and the inference corresponding to SOUL and NCE models, expressed in number of processed words per second.

sharp differences in translation performance. The use of SOUL CSTMs helps to improve the baseline system by 0.9 BLEU points on the development set, and by 0.7 BLEU points on the test set. With NCE CSTMs, these gains are respectively 0.7 and 0.9 BLEU points. Finally, combining all SOUL and NCE models does not yield further improvement, which indicates that probabilities estimated by SOUL and NCE models are not very different from each other. Their combination does not provide complement information compared to the separate use of each model category.

Table 3.12 compares the training and inference speeds for SOUL and NCE models, which is very in line with the empirical results presented in Table 3.5. While the two kinds of models have the same training speed, in inference the NCE models benefit from their un-normalized scoring.

3.4 Conclusions

In this chapter, we have reviewed two efficient approaches to reduce the cost of computing conditional probabilities when training and integrating CSMs. The first approach consists of employing a hierarchical output layer (the SOUL structure), which organizes the output vocabulary as a clustering tree, and transforms the word prediction into the prediction of a path leading from the root of the tree to one of its leaves. Instead of basing on a model-specific solution, the second approach (NCE) comes from the category of sampling-based methods, which consists of approximating the gradient of the conventional conditional likelihood criterion based on a small subset of negative examples from the output vocabulary.

Two typical sampling-based methods described in this chapter are Biased Importance Sampling (BIS) and Noise Contrastive Estimation (NCE). BIS is based on a biased estimator, which requires to sample a lot of negative examples to approach the estimation to the true distribution, as well as to limit the variance. As the number of sampled negative words cannot be too high, an workaround consists of adapting both this number and the proposal distribution. Compared to BIS, NCE represents a more elegant approach derived from the standard unbiased version of the Importance Sampling, which does not require any adaptation during the training process. Another advantage of the NCE algorithm lies in the fact that this method leads to *self-normalization*, which allows us to also ignore the expensive normalization during the inference.

Experiments in this chapter have helped to empirically compare the trainings (for both monolingual and bilingual CSMs) with SOUL structure, and with NCE criterion on translation tasks from 3 different domains. Several general conclusions can be drawn from this set of experiments. Firstly, SOUL models, which have been trained directly to optimize the conditional likelihood on training data, are generally better in terms of perplexity; but these perplexity differences are often insufficient to lead to sharp improvements in translation performance. Secondly, the un-normalization feature of NCE models is not a weakness of this kind of models, and the *self-normalization* deduced from the NCE optimization is sufficient for the model integration into SMT systems. Moreover, this un-normalized scoring also helps NCE models to speed up the inference, which results in faster computations compared to those using SOUL models.

Among the experiments with NCE training, we insist particularly on the importance of choosing an appropriate noise distribution. While the unigram distribution is unanimously chosen because of its simplicity, a possible problem that may arise is the inconsistency between this noise and the data distributions. As clearly suggested by the second experiment, there is in practice *no* universal unigram distribution that can be used for all NCE trainings. Instead, it is important that the noise distribution fits the unigram distribution estimated on the training dataset used to train the CSM. This conclusion would be meaningful in large-scale learning techniques where various large corpora of different genres and domains are mixed together, and from which training data is resampled at each iteration in order to reduce the training cost. In such case, it is important to perform data resampling *before* estimating the noise distribution, as suggested by Algorithm 4.

4

Adaptive strategies for learning rates

The high computational cost of training NNLMs is an important issue for a lot of applications. In Section 2.3.4, we have reviewed some possible workarounds to reduce this cost, such as the use of *short-list* (Bengio et al., 2003a; Schwenk and Gauvain, 2004), the hierarchical structuring of output vocabularies (Morin and Bengio, 2005; Mnih and Hinton, 2008; Le et al., 2011), or the use of self-normalized output layers (Vaswani et al., 2013; Devlin et al., 2014). In Chapter 3, we have described two approaches to efficiently train a NNLM, which drastically reduce the training time of each iteration. However, in practice the total training cost also depends on the number of iterations which reflects the convergence speed of each method.

Even though differing in optimization steps, SOUL and NCE models both rely on the Stochastic Gradient Descent (SGD), for which hyper-parameters, such as learning rates, have a great impact on the final convergence level, as well as the time needed to achieve this convergence. In practice, the training of NNLMs uses learning rates which are either empirically determined or adapted. The total training cost must take into account the number of times these hyper-parameter values are tested in order to obtain the optimal performance. An adaptive regime may be helpful, as it aims to automatically achieve the optimal values without the need to repeat in multiple instances the training procedure.

Recently, *Adaptive Gradient* (or AdaGrad (Duchi et al., 2011)) has been used to adapt one learning rate for each model parameter. In the context of training NNLMs, it requires however additional cost to maintain a high-dimensional vector of learning rates. Moreover, the method still depends on other hyper-parameters which need to be empirically set.

Convergence speed is an important issue for the NCE training, which has been experimentally shown to be less stable than the maximum-likelihood estimation (Chapter 3). The NCE training introduces an additional random element which is the sampling from the noise distribution. Moreover, the number of updates performed on the word embed-

dings may differ greatly from a word to another depending on the noise distribution; the parameters corresponding to rare words may not be updated at all or obtain very few updates. Given this context, a careful choice of an adaptive scenario for learning rates offers a promising and simple technique to reduce the number of iterations needed and the computational cost without changing the general training method.

In this chapter, we propose to compare several methods for adapting the learning rates of the SOUL and NCE trainings. Two new methods are introduced which aim at (a) reducing the dependence of the performance on hyper-parameters, and (b) subtly adjusting one learning rate for each group of parameters in order to guide the model to achieve a faster convergence. Between these two goals, we emphasize particularly on the first one as the empirical hyper-parameter fixing is particularly time-consuming. The next section briefly reviews current methods for adapting learning rates in the SGD training framework. Note that this review is far from exhaustive; we focus only on methods that can be applied to our model structure. Several experimental comparisons in a practical large-scale setting will be described in Section 4.3; the experimental set-up is based on an English-to-Spanish SMT task. Results related to the training of SOUL NNLMs have been published in (Do et al., 2014b). In this chapter, we extend this work with additional experiments on the NCE training of NNLMs.

4.1 Stochastic Gradient Descent, and beyond

The conventional NNLM is trained by minimizing an empirical regularized log-likelihood criterion on a dataset $\mathcal{S}_n$:

$$\mathcal{L}_{cll}(\boldsymbol{\theta}, \mathcal{S}_n) = \sum_{(w,\mathbf{c})\in\mathcal{S}_n} -\log \mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c}) + \mathcal{R}(\boldsymbol{\theta}) \tag{4.1}$$

where, as in previous chapters, $\boldsymbol{\theta}$ denotes the vector containing all free parameters of the CSM, and $\mathbf{p}_{\boldsymbol{\theta}}(.)$ is the probability distribution computed by Equation (2.12). The dataset $\mathcal{S}_n$ is viewed as a set of n-grams, or more generally as pairs of a word w and a context $\mathbf{c}$ from which w is predicted.

The use of the criterion (4.1) requires model scores to be normalized over the output vocabulary $\mathcal{V}^o$. This normalization represents however a major computational difficulty for training as well as inference of an NNLM, as has been discussed in Section 2.3.4. In the previous chapter, we have described a method which consists of using a structured output layer (Section 3.1); in this case the training objective function (4.1) is kept unchanged. Another approach is to use the NCE algorithm (Section 3.2) which optimizes the corresponding NCE criterion :

$$\mathcal{L}_{nce}(\boldsymbol{\theta}, \mathcal{S}_n) = \sum_{(w,\mathbf{c})\in\mathcal{S}_n} \left[-\log \frac{\mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c})}{\mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c}) + K\mathbf{p}_N(w)} - \sum_{i=1}^{K} \log \frac{K\mathbf{p}_N(\hat{w}_i)}{\mathbf{p}_{\boldsymbol{\theta}}(\hat{w}_i|\mathbf{c}) + K\mathbf{p}_N(\hat{w}_i)} \right] + \mathcal{R}(\boldsymbol{\theta}) \tag{4.2}$$

where $\hat{w}_{i=1}^{K}$ are K negative examples randomly generated from a noise distribution $\mathbf{p}_N(.)$. In both cases, the objective function is coupled with a $\mathcal{L}_2$-regularization term $\mathcal{R}(\boldsymbol{\theta}) = \gamma \times \frac{||\boldsymbol{\theta}||^2}{2}$.

Though differing in training criteria, both approaches typically use SGD algorithm to optimize $\boldsymbol{\theta}$, which consists of iteratively looking for a better $\boldsymbol{\theta}^{(t+1)}$ at time $t+1$ given its current value at time t :

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}^{(t)}, \mathcal{S}_n) \tag{4.3}$$

where $\nabla \mathcal{L}(\boldsymbol{\theta}^{(t)}, \mathcal{S}_n)$ denotes the gradient at the current value $\boldsymbol{\theta}^{(t)}$, and η is the learning rate. Without loss of generality, we assume here that η is a vector which contains all learning rates, one for each parameter; however η may turn out to be a scalar if all of its components take the same value.

When the mini-batch SGD algorithm is used to train a NNLM (Schwenk and Gauvain, 2004; Bengio, 2012), we estimate the gradient, then update $\boldsymbol{\theta}$ after each mini-batch [1], instead of waiting until the whole dataset is used. This stochastic method is theoretically proved to be worse than the batch method. In practice, however, it seems to be more convenient than the batch version as it leads to a faster convergence while using fewer data. The size of these mini-batches is set so that it allows us to efficiently use matrix-matrix operations which are generally easier to optimize than vector-matrix operations.

The simple update form (4.3) however yields small incremental changes on deeper neural networks with many hidden layers (Martens, 2010). The reason is that the optimization direction estimated based on the gradient tends to ignore information about the curvature of the objective function and does not allow the training to *accelerate* in regions where both the objective value and its gradient change slowly [2] [3]. This curvature information is expressed not in the gradient itself, but in the second-order *gradient of gradient*, or Hessian matrix. There are henceforth some attempts to use Hessian-like algorithms or conjugate gradient approaches to train a neural network. Originally, Newton algorithm replaces the learning rate η in (4.3) by the inverse of the Hessian matrix estimated at time t :

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \left(\mathbf{H}^{(t)}\right)^{-1} \nabla \mathcal{L}(\boldsymbol{\theta}^{(t)}, \mathcal{S}_n)$$

where $\mathbf{H}^{(t)}$ is the Hessian matrix computed at $\boldsymbol{\theta}^{(t)}$. This method is constrained by the high cost of storing the Hessian matrix and computing its inverse, especially in large neural networks. Indeed, the Hessian matrix's size is $|\boldsymbol{\theta}|^2$, with $|\boldsymbol{\theta}|$ denoting the number of network parameters. A typical NNLM structure contains a lookup table (Section 2.3.1) of 500 dimensions for each of 372K vocabulary words; the total number of trainable parameters may be up to 2×10^8. Therefore, some *Quasi-Newton* methods propose to approximate $\mathbf{H}^{(t)}$. A review of these methods for neural networks can be found in (LeCun et al., 2012; Bengio, 2012). A plausible approach is to use a diagonal approximation to $\mathbf{H}^{(t)}$, which is equivalent to the use of a separate learning rate for each parameter in the model. This approach will be further described in the next section.

Another solution is Hessian-free optimization procedures which have also been developed to overcome the difficulties associated to the training of complicated neural structures. The main idea is that, instead of estimating and storing $\mathbf{H}^{(t)}$, one should only

[1] Each mini-batch contains some hundreds training examples, typically 128.

[2] These regions are called to exhibit *pathological curvature* in (Martens, 2010).

[3] An example of this phenomenon is the *vanishing gradient* problem in training recurrent NNs (Section 2.3.2).

compute $\mathbf{H}^{(t)} \times \mathbf{d}$ where $\mathbf{d}$ is a $|\boldsymbol{\theta}|$-dimensional vector. Fortunately, this matrix-vector product can be efficiently estimated using finite differences at the cost of a single additional gradient evaluation which, in case of neural networks, corresponds to an additional forward-backward step :

$$\mathbf{H}^{(t)}\mathbf{d} = \lim_{\epsilon \to 0} \frac{\nabla \mathcal{L}(\boldsymbol{\theta}^{(t)} + \epsilon \mathbf{d}) - \nabla \mathcal{L}(\boldsymbol{\theta}^{(t)})}{\epsilon}$$

Secondly, Hessian-free approaches employ *conjugate gradient* optimization to find an optimal update direction from $\boldsymbol{\theta}^{(t)}$ to $\boldsymbol{\theta}^{(t+1)}$. Martens (2010) and Martens and Sutskever (2011) develop an improved variant of the Hessian-Free optimization which is powerful enough to train deep neural networks from random initializations. This approach has been successfully applied to train a character-level NNLM (Sutskever et al., 2011), and is shown to be an effective solution to the vanishing gradient associated with recurrent NNLMs (Section 2.3.2). However, the Hessian-free method as described in (Martens, 2010) is still not free from hyper-parameters. Moreover, for feed-forward neural structures applied to language modelling task, the plain SGD is much simpler to implement but still effective in terms of performance as well as of computational cost, while an appropriate learning rate schedule helps to improve training speed and precision. This is the approach we adopt for the rest of this chapter.

4.2 Adaptive strategies for the learning rate

In most cases, NNLMs use a learning rate that is empirically fixed or adapted by some simple regimes. To the best of our knowledge, there is still no comparison between different regimes to assess how they could affect the performance as well as the training time of a NNLM. However, in the Stochastic Gradient Descent literature, the learning rate is considered since early works as a crucial hyper-parameter that greatly impacts the convergence rate (Robbins and Monro, 1951). More importantly, a good adaptive learning rate schedule, although without much tuning, seems to perform equivalently or even better than the best-tuned SGD, hence eliminates the need for manual tuning of the learning rate (Schaul et al., 2012). These adaptive methods can be divided into two groups, those using a global learning rate, and those adjusting a separate learning rate for each parameter, or each group of parameters.

4.2.1 A global learning rate

The most widely used global learning rate schedule is the **Power Schedule** $\eta \approx t^{-1}$ described in (Senior et al., 2013). This regime is theoretically proven to give the best asymptotic convergence in the case of a global learning rate (Xu, 2011) [4]. More precisely, the learning rate $\eta^{(t)}$ at time t is decayed according to the number of updates performed

[4]The authors in (Xu, 2011) use an average version of SGD while we still choose the best iteration according to the objective value on held-out data.

on parameters as :

$$\eta^{(t)} = \frac{\eta^{(0)}}{1 + \tau t}$$

where τ is a learning rate decay coefficient. In (Bengio et al., 2003a), τ is empirically set to 10^{-8}. Le et al. (2012b) use a special form of **Power Schedule** in which :

$$\eta^{(t)} = \frac{\eta^{(0)}}{1 + \tau \mathrm{N}_e^{(t)}}$$

where $\mathrm{N}_e^{(t)}$ denotes the number of training examples used until the time t. This is the form we adopt in our experiments described in Section 4.3.

Besides the Power Scheduling, another regime named **Down Schedule** is introduced for training the SOUL structure (Le, 2012). The scenario consists of dividing the training process into 2 periods. During the first one, the learning rate is kept fixed; in the second one, this quantity is divided by 2 after each iteration. [5] The boundary between these two periods is identified for instance by monitoring the perplexity on a held-out (or validation) set : it can be defined as the moment when perplexity starts to increase. A variant of this method named **Adjust Schedule** (used for example in (Ollivier, 2013)) consists in comparing the variation of perplexity between two consecutive iterations : if it increases, we cancel all the updates of the current iteration and divide the learning rate by 2. Conversely, if a decrease is observed, we multiply the learning rate by 1.1. This schedule practically allows us, as training goes on, to have a learning rate reasonably independent from its initial value and adapted according to an intrinsic measure. However, some iterations are wasted; this phenomenon appears more frequently during the NCE training which is less stable than the maximum-likelihood estimation with SOUL structure.

The simplest method among all is the **Fix Schedule** that keeps the learning rate constant during all the training. In our work, this regime serves mainly as a reference to measure how important a good adaptive learning rate schedule is for the final convergence and the training time.

4.2.2 Local learning rates

Reviews on training methods for neural networks often recommend the use of different learning rates for different parameters located in various parts of the network (Bottou, 2012; LeCun et al., 2012). This approach can be considered to be the intermediary between standard SGD and Hessian-like methods such as Gauss-Newton or Quasi-Newton algorithms, with the Hessian-like matrix reduced to its diagonal approximation. For instance, LeCun et al. (2012) recommend the use of diagonal Gauss-Newton and diagonal Levenberg-Marquardt algorithms for classification problems.

Recently, *Adaptive Gradient* (or AdaGrad (Duchi et al., 2011)) has emerged and has been applied to some NLP applications where it helps to give good performance

[5] no matter whether the perplexity increases or decreases.

(e.g. (Green et al., 2013)). In its original form, AdaGrad requires to approximate the Hessian matrix in Newton algorithm at time t as follows :

$$\mathbf{H}^{(t)} \approx \left(\sum_{i=1}^{t} \nabla \mathcal{L}(\boldsymbol{\theta}^{(i-1)}) \nabla \mathcal{L}(\boldsymbol{\theta}^{(i-1)})^T \right)^{1/2}$$

which is intractable in case of a large-dimension parameter vector $\boldsymbol{\theta}$. Henceforth, AdaGrad is more often used in its diagonal form which is computationally cheaper. More precisely, let $\mathbf{g}_j^{(i)}$ be the gradient of the loss function with respect to the parameter $\boldsymbol{\theta}_j^{(i)}$ at time t, then the learning rate corresponding to the parameter $\boldsymbol{\theta}_j$ is computed as follows :

$$\eta_j^{(t)} = \frac{\eta^{(0)}}{\left(1 + G_j^{(t)}\right)^{1/2}} \tag{4.4}$$
$$G_j^{(t)} = \sum_{i=1}^{t} \left(\mathbf{g}_j^{(i)}\right)^2$$

where η^0 is the initial learning rate (for all parameters), and the unity is added in the denominator of (4.4) to assign the learning rates at time 0 to $\eta^{(0)}$.

4.2.3 Block-AdaGrad strategy

In our experiments, it is preferable not to use directly (4.4) due to its computational costs. Indeed, the implementations of NNLMs often come up with an intensive use of matrix-matrix and matrix-vector product operations, especially in parameter updating. Consider an update according to (4.3) of the hidden layer weight matrix $\mathbf{W}^h$. The gradient $\nabla \mathcal{L}(\boldsymbol{\theta})$ is obtained from a matrix-vector product in case of the original SGD training, and from a matrix-matrix product if the mini-batch version is used. Multiplying the gradient with η and subtracting the resulted quantity from $\boldsymbol{\theta}^{(t)}$, the formula (4.3) can be performed entirely in one single step [6] only if η is a scalar. Otherwise, the implementation would require first to estimate and store the gradient, then to incorporate the vector η to this gradient via dot products, and finally to subtract the whole quantity from $\boldsymbol{\theta}^{(t)}$. The procedure is switched from one-step to three-step operation and increases the running time of each iteration. The total amount of training time might not be reduced even if fewer iterations are needed to achieve convergence.

Instead of adjusting one learning rate for each free parameter, we prefer using one scalar learning rate whenever the update formula (4.3) is used. In our SOUL and NCE implementation, the update steps are performed by blocks of parameters; it is hence preferable to assign one learning rate to each block. This procedure, adapted from AdaGrad, will be called **Block-AdaGrad** in our experiments.

More precisely, the local learning rate η_j in (4.4) is replaced by η_{b_j} which denotes the learning rate assigned to all parameters belonging to a block b_j. The corresponding

[6] using multi-thread BLAS matrix operation libraries.

procedure is written as :

$$\eta_{b_j}^{(t)} = \frac{\eta^{(0)}}{\left(1 + G_{b_j}^{(t)}\right)^{1/2}} \tag{4.5}$$

$$G_{b_j}^{(t)} = \sum_{i=1}^{t} \left(\mathbf{g}_{b_j}^{(i)}\right)^2 = \frac{1}{|b_j|} \sum_{i=1}^{t} \sum_{k \in b_j} \left(\mathbf{g}_k^{(i)}\right)^2 \tag{4.6}$$

The modification brought to (4.6) is interpreted as follows : at each update, we first modify the value of G_{b_j} corresponding to the block b_j in consideration, by adding to it the square of gradients averaged on all parameters in the block. The resulted G_{b_j} is used to compute the effective learning rate (Equation (4.5)) which will then be used to update all parameters in b_j.

Parameter blocks in SOUL and NCE models : From implementation details, parameter blocks are defined in a manner such that each of them is updated using one matrix-matrix or matrix-vector operation. In SOUL, two blocks are defined for each hidden layer and for each output linear softmax layer : one for the weight matrix and the other for the bias vector. For the lookup table, we have one block for the embedding vector of each vocabulary word. The Block-AdaGrad is designed to introduce no extra computing time compared to the standard method. The total number of blocks amounts to 376K for a total of 378 millions of parameters.

NCE models use a conventional output layer which is organized and trained similarly to the lookup table layer. [7] Therefore, each word has its embedding updated with a separate learning rate.

4.2.4 Down-Block-AdaGrad

In practice, few adaptive scenarios are really free from the choice of hyper-parameters. For the adaptive learning rate methods described above, a hyper-parameter which is shown to play an important role (Section 4.3.2) is the initial value $\eta^{(0)}$, except for **Power Scheduling** where the performance may depend not only on $\eta^{(0)}$ but also on the learning rate decay coefficient τ. The strong dependence on hyper-parameters of some methods such as Block-AdaGrad, as will be shown in Section 4.3.2, is an unexpected property, as it implies we need to repeat the training procedure with different hyper-parameter values in order to obtain the optimal performance. This choice leads inevitably to extra computational cost. Moreover, the more the performance depends on hyper-parameter values, the more it becomes important to perform the corresponding grid-based search. On the other hand, if an adaptive method is proven to depend only slightly on hyper-parameters within a reasonably large range of values, it maybe suffices to choose an arbitrary value inside this range without a much more careful search. The dependence on hyper-parameters is henceforth a motivation behind some recent AdaGrad variants such as AdaDec (Senior et al., 2013) and AdaDelta (Zeiler, 2012) which propose strategies to reduce the sensitivity of their methods to hyper-parameters.

[7] The parameters of this layer can be considered as output word embeddings in the standard NNLM of (Bengio et al., 2001, 2003a).

In our study with SOUL and NCE models, we propose a simpler approach by combining *Block-AdaGrad* with a global learning rate adaptive regime. This idea is derived from the observation (Section 4.3.2) that, when training SOUL models, the *Down Scheduling* depends only slightly on the initial value $\eta^{(0)}$. More precisely, on top of the *Block-AdaGrad*, we adjust an extra global scaling coefficient $\gamma^{(t)}$ such that at every moments during the training process, the learning rate of a reference parameter block b^* is scaled to be equal to $\gamma^{(t)}$. For the learning rates of blocks other than b^*, the ratio between any learning rate and the one of b^* must be kept the same as in the *Block-AdaGrad* method. We name this method *Down-Block-AdaGrad*, which adjusts the learning rate $\eta_{b_j}^{(t)}$ assigned to block b_j at time t by the following formula :

$$\eta_{b_j}^{(t)} = \frac{\gamma^{(t)} \times \left(1 + G_{b^*}^{(t)}\right)^{1/2}}{\left(1 + G_{b_j}^{(t)}\right)^{1/2}} \tag{4.7}$$

where b^* denotes the reference block which, in SOUL and NCE models, is assigned to the block of weight matrix of the first output layer. [8] This choice is motivated by the fact that perplexity is particularly sensitive to the last layer of the network. The parameter $\gamma^{(t)}$ is in turn adjusted using *Down Scheduling* with $\eta^{(0)}$ as its initial value.

Why should this strategy work? On the one hand, the method still benefits from a local adjustment which is an advantage of *Block-AdaGrad* : the vector of learning rates is assumed to approximate the Hessian matrix, and to give information about the curvature of the loss function. Here the ratio between any pair of learning rates inside *Down-Block-AdaGrad* is kept the same as in *Block-AdaGrad*. On the other hand, the method improves a weakness of *Block-AdaGrad* that its learning rates may be decayed too fast before the training converges. This strategy will be compared on the same ground along with other adaptive scenarios described above in the large-scale training of a NNLM included in a SMT system.

4.2.5 Peculiarities of NCE training

As mentioned above, the *Block-AdaGrad* applied on NCE models differs from the one on SOUL models by the fact that the output layer's parameters are grouped in different parameter blocks, one for each output vocabulary word. This arrangement is after all a practical choice, decided based on the feature of the NCE implementation. Besides this, our preliminary experiments suggest that the learning rates corresponding to the output layer are decayed too fast. Hence, we modify Equation (4.6) as follows :

$$G_{b_j}^{(t)} = \frac{1}{|b_j| \times |\mathcal{V}^o|^{1/2}} \sum_{i=1}^{t} \sum_{k \in b_j} \left(\mathbf{g}_k^{(i)}\right)^2 \tag{4.8}$$

where $|\mathcal{V}^o|$ is the size of the output vocabulary. Note however that this modification intervenes only in the NCE training, and only in the output layer. This also modifies accordingly the *Down-Block-AdaGrad* scenario.

[8] It corresponds to the principal output layer in SOUL structure, while in NCE model it is the unique output layer.

It is also observed that training with the NCE algorithm is much less stable than the maximum-likelihood estimation. Here this *instability* means that the outcome of each iteration depends heavily on the sampling of negative examples (Section 3.2), and that the perplexity is not guaranteed to decrease after each iteration. In order to ensure that the NCE training will finally converge, we apply a simple trick from *Adjust Scheduling* which consists of cancelling all updates and getting back to the previous parameter values if the current perplexity is found not to decrease after the current iteration. For the NCE training, this technique is systematically applied no matter what adaptive regime is used.

4.3 Experiments

4.3.1 Description of experiments and methodology

To assess the impact of the different learning rate schedules on NNLMs, we carry out experiments on the Spanish language modelling task to compare different scenarios described in the previous section : the global learning rate methods (*Fix, Power, Down*, and *Adjust Schedules*), along with the two variants of AdaGrad (*Block-AdaGrad* and *Down-Block-AdaGrad*). Two kinds of training, which have been described in Chapter 3, are used in this comparative study : the maximization of the conditional likelihood described in Section 3.1, and the NCE algorithm trained on the standard structure (Section 3.2). NNLMs are designed for the large-scale English-to-Spanish shared translation task at WMT'2013. [9] These Spanish NNLMs are 10-gram models which are trained using the same vocabulary and datasets used to train the Spanish back-off LMs that were a part of our SMT system submitted to this task (Allauzen et al., 2013). Neural networks are trained iteratively : at each iteration a 15M-ngram subset is sampled from the whole training data, and then divided into mini-batches of 128 n-grams. The held-out data (or validation set) is *Newstest2008*. As described in (Allauzen et al., 2013), the training scheme of the standard back-off LM is decomposed as follows : the total training data is divided into 7 sets based on dates or genres; for each set, a standard 4-gram LM is estimated using modified Kneser-Ney Smoothing (Chen and Goodman, 1996). These LMs are then interpolated using coefficients chosen so as to minimize the perplexity. These interpolation coefficients are re-used to train the NNLMs. For each configuration and at each iteration, the quantities of data sampled from those 7 sets are proportional to these coefficients. For a fair comparison between methods, data used at all iterations is sampled only once before any training starts, so that all adaptive scenarios will use exactly the same data during their training process. The same technique is also carried out with the sampling of negative examples for NCE training; the sampling is done beforehand and is used for all adaptive methods. We use a NNLM structure with 500-dimension lookup table and two hidden layers of 1000 and 500 units in all experiments.

Our study emphasizes particularly the dependence of the methods on hyper-parameters, which are the initial learning rate $\eta^{(0)}$ and the learning rate decay τ in case of *Power Schedule*. The other meta-parameters are fixed to pre-defined values, for instance the fact that we divide the learning rate by 2 if the perplexity increases, and that

[9] http://statmt.org/wmt13/translation-task.html .

	Fix	Pow $5e-7$	Dow	Adj	Bag	D-Bag
$5e-03$	93.9	103.8	91.4	91.3	100.9	**90.8**
$1e-02$	97.8	99.7	91.4	91.0	96.4	**90.8**
$2e-02$	104.4	96.2	92.2	**90.1**	92.6	92.9

Table 4.1 – The best perplexities obtained by SOUL models on *Newstest2008* by different adaptive learning rate strategies, corresponding to 3 values of the initial learning rate.

a factor of 1.1 is used to reinforce the learning rate in *Adjust Schedule*. In principle, the choice of hyper-parameter values should be performed by grid-based search in which a series of values are assigned on multiple training instances and final outcomes are compared. The high computational cost of training NNLMs does not allow us to perform a full search over a large number of values for $\eta^{(0)}$, however the following experiments still reflect some general trends. More precisely, SOUL models are trained using each adaptive scenario with 3 values of $\eta^{(0)}$: $0.005, 0.01$ and 0.02 while τ is set to 5×10^{-7} in *Power Schedule*. For the NCE training, a slightly extended search is done with 5 values of $\eta^{(0)}$: $0.005, 0.01, 0.02, 0.05$ and 0.1, while 3 values of τ are tested : $5 \times 10^{-7}, 1 \times 10^{-7}$, and 5×10^{-8}.

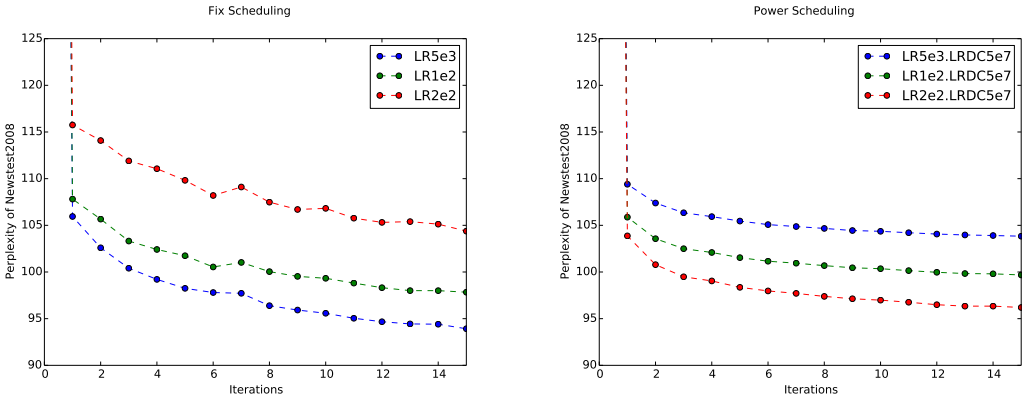
The schedules are evaluated based on different criteria : perplexities estimated on the validation set, as well as the convergence speed and the sensitivity to hyper-parameter values. The held-out dataset is employed after each iteration to re-compute the perplexity. This perplexity serves two purposes : it is an indication used by different schedules to adjust learning rates, while the smallest value at the end of the training process is helpful for us to assess the performance. Convergence speed is mentioned here in its practical meaning : we fix in advance the number of training iterations to 15 [10] and compare the perplexities obtained by different methods. For a theoretical asymptotic analysis on convergence speed of the SGD algorithm, the reader is invited to read, for instance (Duchi et al., 2011). As an extrinsic evaluation, the NNLMs are also included in the SMT system via N-best rescoring to estimate the impact of these adaptive methods on the final translation performance.

4.3.2 Perplexity-based analyses

We first compare the performance of adaptive scenarios based on the perplexities computed on the validation set *Newstest2008* after 15 training iterations. Results of training SOUL models are presented in Figure 4.1a-4.1f and Table 4.1, while the results for NCE models are shown in Figure 4.2a-4.2h and Table 4.2. For the training of SOUL NNLMs, the performance of *Fix*, *Power* and *Block-AdaGrad* schedules is strongly sensitive to $\eta^{(0)}$, whereas *Down* and *Adjust* schedules give a quite consistent results with 3 different values of $\eta^{(0)}$. The last two methods also give a faster convergence to a lower perplexity on the validation set, compared to the *Fix*, *Power* and *Block-AdaGrad* schedules.

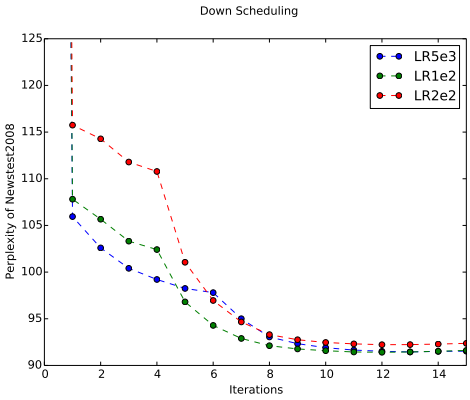
In the case of *Power Schedule*, we also have a complicated problem of choosing τ. It seems that the value of this hyper-parameter has an important impact on the convergence

[10] a value often mentioned in the training of SOUL models (Le, 2012) and which leads to a reasonable amount of training time.

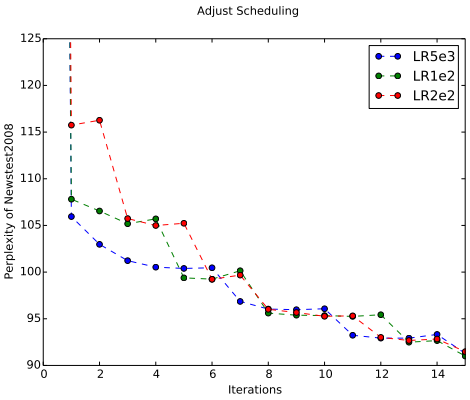


(a) Fix Scheduling

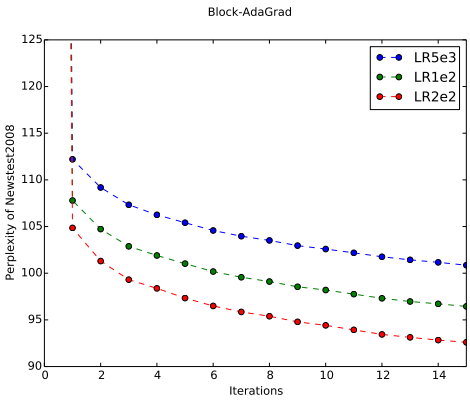
(b) Power Scheduling



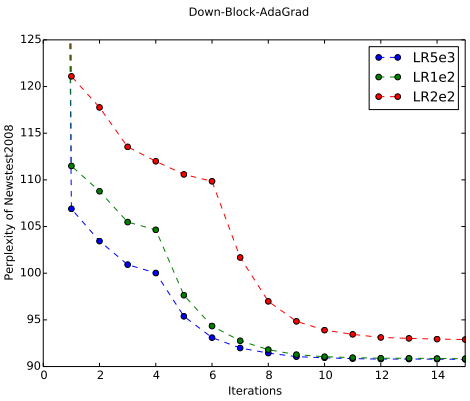
(c) Down Scheduling



(d) Adjust Scheduling

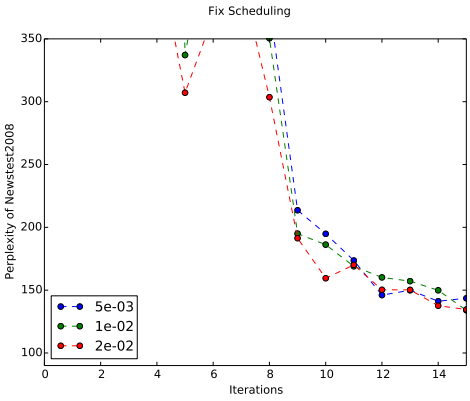


(e) Block-AdaGrad

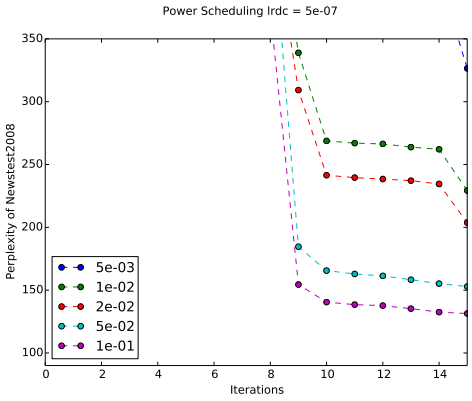


(f) Down-Block-AdaGrad

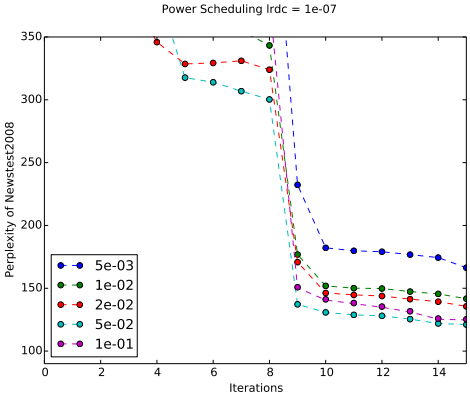
Figure 4.1 – Perplexities computed on *Newstest2008* for each adaptive strategy when training SOUL-structure models.



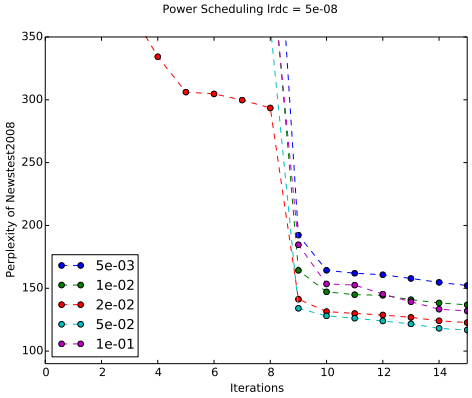
(a) Fix Scheduling



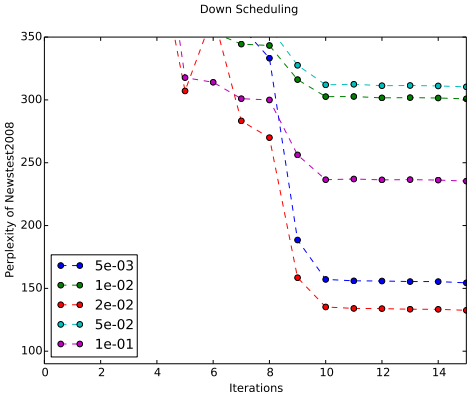
(b) Power Scheduling $\tau = 5 \times 10^{-7}$



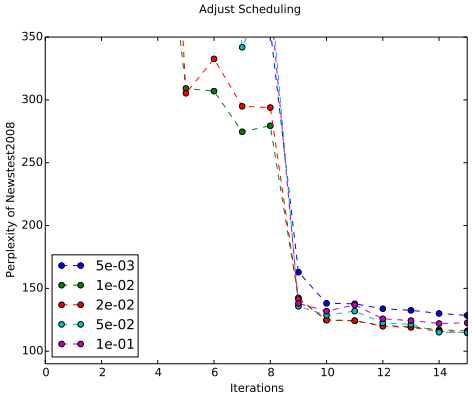
(c) Power Scheduling $\tau = 1 \times 10^{-7}$



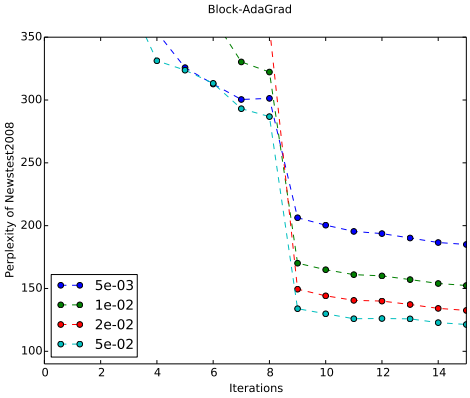
(d) Power Scheduling $\tau = 5 \times 10^{-8}$



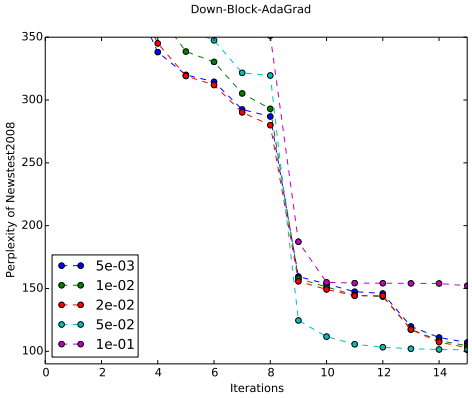
(e) Down Scheduling



(f) Adjust Scheduling



(g) Block-AdaGrad



(h) Down-Block-AdaGrad

Figure 4.2 – Perplexities on *Newstest2008* when training with NCE algorithm.

	Fix	Pow $5e-7$	Pow $1e-7$	Pow $5e-8$	Dow	Adj	Bag	D-Bag
$5e-03$	141.2	326.5	166.4	152.2	154.4	128.5	185.0	**107.0**
$1e-02$	134.0	229.3	141.7	136.8	300.8	116.1	152.4	**104.4**
$2e-02$	134.6	203.7	135.6	122.7	132.5	114.7	132.6	**102.9**
$5e-02$	x	152.9	121.1	116.8	310.4	114.8	121.3	**101.4**
$1e-01$	x	131.4	124.8	132.0	235.4	**122.0**	x	152.2

Table 4.2 – The best perplexities obtained by NCE models on *Newstest2008* using different adaptive learning rate strategies, corresponding to 5 values of the initial learning rate. Notation "x" signifies that the NCE training fails to reduce the perplexity with the corresponding configuration.

speed as well as the final perplexity. Our experiments here do not allow us to have a clear comprehension on this dependence. However, the dependence on not only one, but two hyper-parameters is a serious limitation which makes its set-up complicated and time-consuming.

Compared to *Block-AdaGrad*, the proposed *Down-Block-AdaGrad* clearly reduces the sensitivity of the performance to $\eta^{(0)}$. Moreover, the method also helps to achieve the best results : it gives the best perplexities for 2 out of 3 values of the initial learning rate. In summary, if we take the perplexity obtained by the simplest *Fix Schedule* as reference, a good adaptive learning rate schedule can reduce the perplexity on *Newstest2008* by at least 4% and up to 14%.

Our analysis focusses however on NCE training. As the NCE criterion is an approximation of the conditional log-likelihood (Chapter 3), NCE training is known to be less stable than the training with SOUL, partly due to the random sampling of K negative samples. In this context, an appropriate learning rate regime appears to be very important. Indeed, a too high initial value may lead to the training failing to reduce the perplexity (Table 4.2). Like in the case of the maximum-likelihood estimation, NCE training shows a high sensitivity of *Fix*, *Power* and *Block-AdaGrad* schedules to $\eta^{(0)}$. [11] But unlike the training of SOUL models, NCE training displays a degradation for *Down Schedule*. This limitation is due to the fact that the second period (see Section 4.2.1) may come too early, as each NCE training iteration is less predictable and yields an increased perplexity more often than the MLE. As a consequence, the learning rate is so strongly decayed after some iterations that it appears to be too small. As a result, for training NNLMs using the NCE algorithm, we recommend the use of *Adjust Schedule* for the global learning rate setting, or of *Down-Block-AdaGrad* if the adaptation of one learning rate for each update block is required. The proposed *Down-Block-AdaGrad* improves by about 11.6% over the best global learning rate adaptive regime. Compared to the reference of *Fix Schedule*, this method reduces the perplexity on *Newstest2008* by at least 24% and up to 28%.

	Perplexity *Nt08*	dev *Nt11*	*Nt12*	*Nt13*
Without NNLM	-	32.3	32.8	28.7
Fix $\eta^{(0)} = 0.02$	104.4	32.8	33.3	29.0
Fix $\eta^{(0)} = 0.01$	97.8	32.9	33.5	**29.3**
Fix $\eta^{(0)} = 0.005$	93.9	33.0	33.5	**29.3**
Down $\eta^{(0)} = 0.005$	91.4	33.0	33.6	**29.3**
Adjust $\eta^{(0)} = 0.01$	91.0	**33.1**	33.6	29.1
Down-Bloc-AdaGrad $\eta^{(0)} = 0.005$	**90.8**	33.0	33.6	29.2
Down-Block-AdaGrad $\eta^{(0)} = 0.01$	**90.8**	**33.1**	**33.7**	**29.3**

Table 4.3 – BLEU scores using SOUL models on *Newstest2012* and *Newstest2013*.

	Perplexity *Nt08*	dev *Nt11*	*Nt12*	*Nt13*
Without NNLM	-	32.3	32.8	28.7
Fix $\eta^{(0)} = 0.005$	141.2	32.5	33.3	28.9
Fix $\eta^{(0)} = 0.01$	134.0	32.6	33.2	28.8
Fix $\eta^{(0)} = 0.02$	134.6	32.6	33.1	28.8
Adjust $\eta^{(0)} = 0.1$	122.0	32.6	33.0	28.8
Down-Bloc-AdaGrad $\eta^{(0)} = 0.005$	107.0	32.8	33.4	29.0
Down-Bloc-AdaGrad $\eta^{(0)} = 0.01$	104.4	**32.9**	33.3	28.8
Down-Bloc-AdaGrad $\eta^{(0)} = 0.02$	102.9	**32.9**	**33.5**	**29.2**
Down-Bloc-AdaGrad $\eta^{(0)} = 0.05$	**101.4**	32.8	**33.5**	**29.2**

Table 4.4 – BLEU scores using NCE models on *Newstest2012* and *Newstest2013*.

4.3.3 Impacts on the translation performance

We use the SOUL and NCE NNLMs trained with different adaptive learning rate regimes to rescore N-best lists produced by our submitted English-to-Spanish SMT system (Allauzen et al., 2013). We use *Newstest2011* as the development set on which log-linear coefficients are tuned (Section 1.3.1), while *Newstest2012* and *Newstest2013* serve as test sets. We use the batch N-best version of MIRA tuning algorithm described in (Cherry and Foster, 2012) and as implemented in MOSES. [12] The tuning algorithm is run in 8 instances from different initial points, the obtained BLEU scores are then averaged. The results are presented in Tables 4.3 and 4.4. To summarize, the N-best rescoring using our NNLMs improves the baseline system by 0.8 BLEU points on the development set, and respectively 0.9 and 0.6 BLEU points on *Newstest2012* and *Newstest2013*. The NCE models display similar performances as SOUL models, with at most 0.2-point differences on the development and test sets, which is in line with our observations in Chapter 3.

For a comparison between different adaptive methods, the performances of SOUL models show no significant difference between *Down-Block-AdaGrad* and *Fix Schedule*. This is in line with a certain number of past works related to the development of language models in NLP applications, in which a reduction of less than 10% (as observed in our

[11] For the first 3 initial learning rate's values, *Fix Schedule* results in quite similar perplexities, however it fails with the two other values.

[12] http://www.statmt.org/moses/ .

experiments) would not translate to an important and consistent gain of the final performance (Rosenfeld, 2000). On the other hand, NCE training shows more divergence between adaptive regimes. The experiments show that our proposed *Down-Block-AdaGrad* adaptive method can improve over scenarios in which no particular adaptive regime is required (the *Fix Schedule*) by 0.3 BLEU points on the development and test sets.

4.4 Conclusions

In this chapter, we have presented an empirical study of several adaptive learning rate regimes that can be used in the training of NNLMs in order to improve the convergence speed and performance. The comparison is based on the two efficient training strategies that have been described in Chapter 3.

A particularity of this work lies in the use of *AdaGrad* for which we propose a variant that adjusts one learning rate for each parameter block (*Block-AdaGrad*). These blocks are defined in a manner such that each of them is updated using one matrix-matrix or matrix-vector product operation, hence the adaptive regime does not introduce any extra cost compared to the standard training process. Another peculiar feature is our insistence on rendering adaptive scenarios independent from hyper-parameters, such as the initial learning rate value or decay coefficient. Indeed, each hyper-parameter requires supplement effort to empirically optimize its value. This study is meaningful as training NNLMs is still expensive, even with efficient approaches proposed in Chapter 3.

Experiments have been carried out on the Spanish language modelling task for a WMT'2013 English-to-Spanish SMT system, using SOUL and NCE NNLMs. These experiments have shown that the performances of *Fix*, *Power* and *Block-AdaGrad* schedules (Sections 4.2.1 and 4.2.3) are very sensitive to their hyper-parameters, whereas the *Down* and *Adjust* schedules give quite consistent results with different hyper-parameter values (Figures 4.1a- 4.1f and 4.2a- 4.2h). In order to reduce the dependency of *Block-AdaGrad* to the initial learning rate value, we propose to combine it with the *Down* schedule. The resulting *Down-Block-AdaGrad* achieves the best performance (in terms of perplexity) in various trainings of SOUL and NCE models.

In terms of evaluation measures, the work provides another evidence that the perplexity has only a loose relationship with the translation performance. For the training of SOUL models, perplexity divergence due to different adaptive learning rate methods does not lead to significant improvement in terms of BLEU score. On the other hand, the NCE training represents a situation in which these adaptive scenarios lead to more significant performance difference, up to 0.3 BLEU points. Finally, we recommend the use of *Adjust* and *Down-Block-AdaGrad* schedules while training NNLMs using the NCE algorithm.

5

Discriminative Training and Adaptation Methods for CSTMs

Training procedures for continuous-space models (CSMs) so far rely on the optimization of the conditional log-likelihood (CLL), or on an approximative form of the CLL gradient (the NCE algorithm). This training criterion, along with the evaluation of CSMs by *perplexity* has, in many cases, only a loose relationship with the performance of the applications into which CSMs are to be integrated. For Speech Recognition systems, it has been claimed (Rosenfeld, 2000) that only a perplexity improvement of 30% or more is significant and can be translated into performance improvement. For SMT systems, we have observed in Chapters 3 and 4 a number of situations in which a difference in terms of perplexity between models does not necessarily lead to an improvement in translation performance. A typical example is the equivalence in terms of BLEU score between SOUL and NCE CSMs reflected in Tables 3.7, 3.9 and 3.11, although SOUL models are often better than NCE models in terms of perplexity. It is hence likely that the CLL training, resulted from an evaluation by perplexity, hinders the impact of CSMs on the SMT systems. To overcome this issue, a discriminative framework will be introduced in this chapter to guide the learning process of CSMs towards improving the performance of the existing system. This framework can incorporate training criteria which closely correlate to MT quality metrics (such as BLEU), and will be first applied to Domain Adaptation situations, before being extended to training situations.

Domain adaptation (DA) is an important and active research topic in Statistical Natural Language Processing (NLP) (Daumé III and Marcu, 2006; Blitzer, 2008), and is often expressed in terms of finding an optimal combination of a small in-domain dataset with large amounts of out-of-domain data. Because of their scarcity, the in-domain corpus gives domain-specific knowledge which is often best used along with general knowledge inferred from large out-of-domain data. To avoid the dilution of domain-specific knowledge, most approaches consider various kinds of data weighting schemes in order to balance

the importance of in-domain *versus* out-of-domain. The use of mixture models is another useful technique in such situations. In such adaptation scenarios, NLP components need to be retrained, entirely or partly to integrate these new samples, which can be very time-consuming, or even unrealistic in many situations. This is especially problematic for SMT systems, that are typically made of multiple statistical models (Chapter 1). With the emergence of CSMs, questions related to DA need to be looked at carefully when such systems need to be adapted to a specific domain. One of the first studies in DA for CSMs has been described in (Le et al., 2012a) where the authors propose to run some additional epochs of back-propagation using in-domain data, on top of CSMs which have already been trained on out-of-domain data. The method is straightforward as it can be considered as an extended training process using the usual maximum-likelihood, but requires however to retrain all system models; moreover the adaptation of the CSMs is performed separately from the other components. The discriminative framework described in this chapter allows us to adopt a simpler strategy which consists of adapting only the CSM using in-domain data, while keeping other models intact.

While an adaptation task requires to adapt the whole system (including the CSM) to a new domain, a training situation learns the CSM parameters on the same data which has been used to train the baseline system. In such situations, a similar discriminative training method can be used as a remedy to the problem related to the gap between training and testing phases described in Chapter 2. A difficulty of such scenario is that the training performance depends strongly on the N-best lists obtained from the training data, and will be explained in Section 5.4.4. Other aspects of the discriminative framework will also be investigated, such as initialization issues, or a comparison between different training criteria.

The main contribution of this work is to propose and investigate discriminative adaptation and training scenarios on top of SOUL and NCE models (Chapter 3) which leads to a tighter integration of these models within SMT systems. The generic procedure is described in the next section, while several training criteria that can be used in the framework will be described in Section 5.2. For each experimental set-up described in Section 5.3, we compare the discriminative framework with the traditional CLL optimization, and assess the performance of different discriminative training criteria. In Section 5.4, we also discuss two other important issues that can have great impact on the system performance : model initialization, as well as the baseline SMT system from which N-best lists are produced. Parts of the experimental results presented in this chapter have been published in (Do et al., 2014a, 2015a,b).

5.1 Discriminative framework for CSMs

As described in Section 2.5.2 of Chapter 2, continuous-space models are in most cases integrated into SMT systems via a post-processing step called N-best rescoring. For each source sentence $\mathbf{s}$, the decoder is assumed to generate an N-best list $\mathbf{H} = \{\mathbf{h}_1, ..., \mathbf{h}_N\}$ of the N top translation hypotheses, which represents a subset of the reachable search

Algorithm 5 Joint optimization procedure for $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$

```
Initialize θ and λ
for each iteration do
    for M mini-batches do                                   ▷ λ is fixed
        Compute the sub-gradient of L(θ, s) for all s in the mini-batch
        Update θ
    end for
    Update λ using dev set                                  ▷ θ is fixed
end for
```

space. Each hypothesis $\mathbf{h}_i = (\mathbf{t}_i, \mathbf{a}_i)$ is associated with a decoding score :

$$F_{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{h}) = \sum_{k=1}^{M} \lambda_k f_k(\mathbf{s}, \mathbf{h})$$

When rescoring with a continuous-space model [1], $F_{\boldsymbol{\lambda}}(.)$ is augmented to also include an additional feature $g_{\boldsymbol{\theta}}(\mathbf{h})$ computed from the CSM. As explained in Chapter 3, $g_{\boldsymbol{\theta}}(\mathbf{h})$ typically corresponds to the log-probability of the hypothesis $\mathbf{h}$, which is computed exactly by a SOUL model, or only approximatively by a *self-normalized* model :

$$g_{\boldsymbol{\theta}}(\mathbf{h}) = \begin{cases} \sum\limits_{(w,\mathbf{c})\in\mathbf{h}} \log \mathbf{p}_{\boldsymbol{\theta}}(w|\mathbf{c}) & \text{for SOUL model} \\ \sum\limits_{(w,\mathbf{c})\in\mathbf{h}} \mathbf{a}_{\boldsymbol{\theta}}(w,\mathbf{c}) & \text{for NCE model} \end{cases} \tag{5.1}$$

where $\boldsymbol{\theta}$ is the vector containing all CSTM free parameters. The new scoring function used in rescoring is then :

$$G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}) = F_{\boldsymbol{\lambda}}(\mathbf{s}, \mathbf{h}) + \lambda_{M+1} g_{\boldsymbol{\theta}}(\mathbf{h}) \tag{5.2}$$

This scoring function depends on the CSTM parameters $\boldsymbol{\theta}$, as well as on the log-linear coefficients $\boldsymbol{\lambda}$. Contrarily to the standard approach which first trains the CSTM, then tunes all the log-linear coefficients (including λ_{M+1}), our proposal requires to *alternatively tune the vector of coefficients* $\boldsymbol{\lambda}$ *and to adapt the parameters* $\boldsymbol{\theta}$: the former uses the development data (Section 1.3.1), while the latter will use a parallel training corpus.

The corresponding optimization procedure splits the training data into fixed-size mini-batches (typically 128 subsequent sentence pairs). As sketched in Algorithm 5, each mini-batch is used to update $\boldsymbol{\theta}$ while keeping $\boldsymbol{\lambda}$ fixed. The log-linear coefficients $\boldsymbol{\lambda}$ are updated every M mini-batches.

It is important to notice that similar algorithms have been adopted in several other studies (He and Deng, 2012; Gao and He, 2013; Gao et al., 2014) for the parameter estimation of phrase translation models (Section 2.4.4). In our study, tuning $\boldsymbol{\lambda}$ is performed using a standard tuning tool, the Batch N-best MIRA algorithm [2] as proposed in (Cherry and Foster, 2012) and described in Section 1.3.1. The update of $\boldsymbol{\theta}$ (with fixed $\boldsymbol{\lambda}$) is more interesting and requires an appropriate training objective function.

[1] The approach can be easily generalized to integrate *more than* one additional model.

[2] We use the implementation included in MOSES toolkit (`http://www.statmt.org/moses/`).

5.1.1 MLE *versus* discriminative training criteria

It is interesting to realize that Algorithm 5 can be performed with any objective function $\mathcal{L}(\boldsymbol{\theta}, \mathbf{s})$, including the conditional log-likelihood (CLL) conventionally optimized to train a NNLM or a CSTM (Section 3.1). The optimization of CLL aims at minimizing the perplexity, which is an intrinsic measure to evaluate a CSM. However, conforming to the experiments already described in Chapters 3 and 4, there are many examples in the literature in which a reduction in perplexity does not translate to a significant improvement of the SMT system.

Evaluating a CSTM throughout its integration within a SMT system, we realize that the consistency between the CLL and the translation performance is loose also because translation outputs do not depend only on the CSTM, but also on other system components, and on the integration mode. If an N-best list rescoring approach is chosen to perform this integration, the final result may also depend on the N-best lists produced by the decoder. A joint training procedure like Algorithm 5 enables to reduce the gap between the training and testing phases. An objective function $\mathcal{L}(\boldsymbol{\theta}, \mathbf{s})$ needs to be designed to further reinforce this correlation. In this context, discriminative tuning algorithms, as described in Section 1.3.1 of Chapter 1 are an important source of inspiration, as they represent two following advantages. First, batch (and certain online) tuning algorithms use intensively N-best lists of hypotheses, which are often the environment in which a CSTM is incorporated. Second, the incorporation of MT evaluation metrics (such as BLEU) into these algorithms has become a standard assumption in SMT since the introduction of MERT (Och, 2003). It is also desirable that such training criterion makes the CSTM training dependent on existing components of the SMT system. As a consequence, we will define (in Section 5.4.3) our discriminative criteria based on the final scoring function $G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h})$ which involves all the system scores, along with the log-linear coefficients $\boldsymbol{\lambda}$.

5.1.2 Discriminative domain adaptation with CSMs

Even though CSMs have been proved to boost the translation performance, the adaptation of CSMs for SMT has received little attention. In the first work on lexicalized CSTM, Le et al. (2012a) present an adaptation experiment in which their SMT system, which already includes an out-of-domain CSTM is *adapted* to a new domain by simply running five additional epochs of the back-propagation algorithm on in-domain data. This simple strategy gives an improvement of 0.7 BLEU points, which shows that continuous-space models in general, and SOUL CSTM in particular, can be efficiently used in a DA setting. However, the experiments in (Le et al., 2012a) still require that each individual model is retrained with the arrival of in-domain training samples. Moreover, the CSTM is adapted separately from other models of the system, meaning that the adaptation process is unaware of potentially important information about the CSTM interaction with other components.

In this study, we adopt a different practice in which all out-of-domain component models are kept the same, only the CSTM needs to be *adapted* using a small in-domain parallel corpus. Figure 5.1 summarizes our experimental practices : the left-hand side shows the conventional 2-step building of a SMT system, while the discriminative adap-

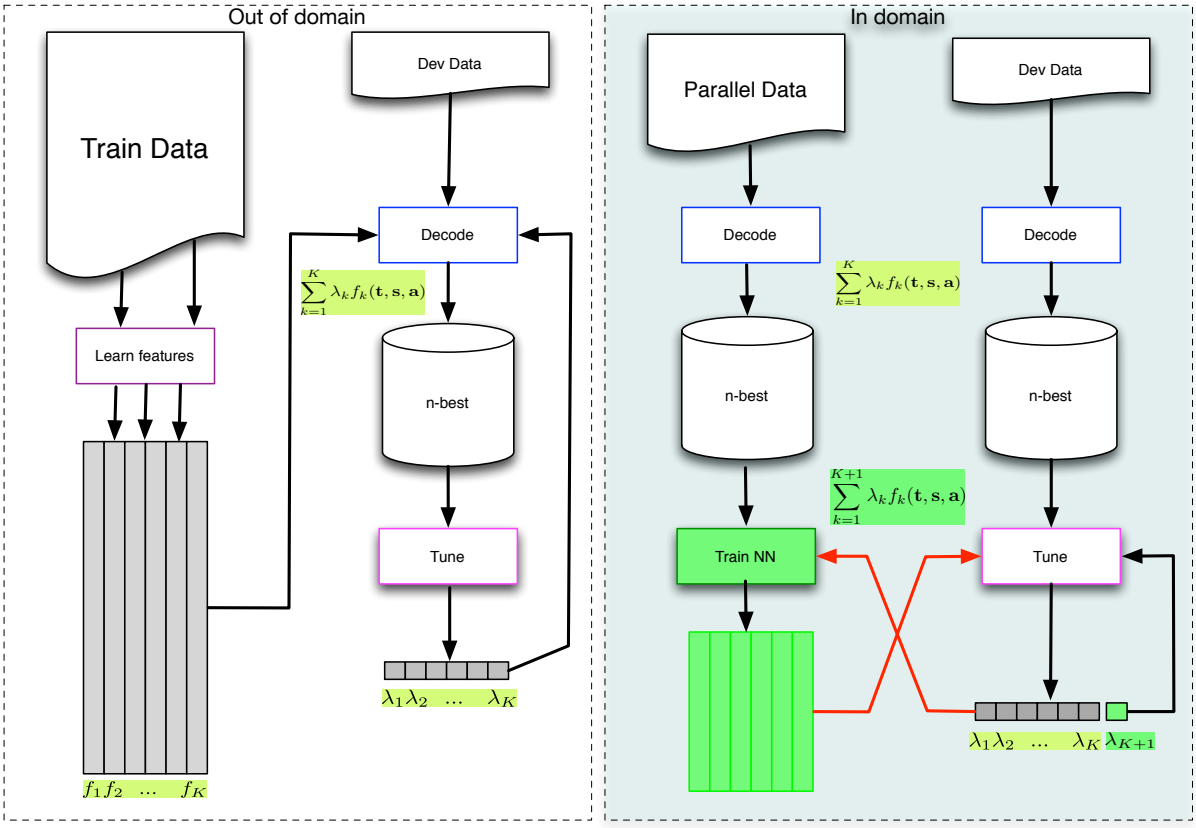


Figure 5.1 – The two-step building of a SMT system (left-hand side) and the discriminative training (right-hand side). In the left, training data is used to separately train each feature model (from which feature scores are given to the log-linear model), then the log-linear coefficients $\boldsymbol{\lambda}$ are tuned on the N-best lists of the development set. On the other hand, discriminative training exploits N-best lists on both training and development data. The training criterion involves the whole scoring function, and alternatively updates feature function parameters on the training data, and tunes $\boldsymbol{\lambda}$ on the development data.

tation scenario is sketched in the right-hand side. The traditional schema is used to build the baseline SMT system (including a CSTM), while the discriminative framework is employed to adapt the CSTM.

5.1.3 Computation of gradient using back-propagation

Stochastic Gradient Descent (SGD) is used to optimize the loss function $\mathcal{L}(\boldsymbol{\theta}, \mathbf{s})$. The dependence of the loss function on $G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h})$ constitutes an additional layer throughout which the gradient with respect to $\boldsymbol{\theta}$ can be computed using the chain rule. This gradient is then used to update $\boldsymbol{\theta}$ while keeping $\boldsymbol{\lambda}$ fixed. More precisely, the computation can be formulated in four steps :

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta}, \mathbf{s})}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{L}(\boldsymbol{\theta}, \mathbf{s})}{\partial G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_1^N)} \times \frac{\partial G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_1^N)}{\partial g_{\boldsymbol{\theta}}(\mathbf{h}_1^N)} \times \frac{\partial g_{\boldsymbol{\theta}}(\mathbf{h}_1^N)}{\partial g_{\boldsymbol{\theta}}((w, \mathbf{c}) \in \mathbf{h}_1^N)} \times \frac{\partial g_{\boldsymbol{\theta}}((w, \mathbf{c}) \in \mathbf{h}_1^N)}{\partial \boldsymbol{\theta}} \tag{5.3}$$

where the first term reflects the dependence of the loss function on the system score $G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)$ of each hypothesis $\mathbf{h}_i \in \{\mathbf{h}_1, ..., \mathbf{h}_N\}$, which depends itself on the definition of the loss function. The second term indicates the integration of model score $g_{\boldsymbol{\theta}}()$ within the system score $G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)$. It is simply $\lambda_{M+1} \times I_N$ where I_N is the identity matrix of dimension N. The term is in practice ignored as it only involves a change in the learning rate. The third term reflects the transition from hypothesis-level to n-gram-level scores estimated by the neural network. This computation will be described below in greater detail. Finally, the fourth term is conventionally computed by back-propagation inside SOUL or NCE models. The computation of this term, along with several approximations has already been described in Chapter 3.

In general, the gradient at (5.3) requires dealing with each n-gram $(w, \mathbf{c})$ extracted from N-best hypotheses. Let T be the number of these n-grams. Given the fact that the log-probability of each hypothesis $\mathbf{h}_i$ is the sum of log-probabilities of its n-grams, the third term in (5.3) is a $N \times T$ matrix $\mathbf{D}$ in which each element $\mathbf{D}_{i,j}$ counts the number of times the j^{th} n-gram appears in $\mathbf{h}_i$. The right-hand side of (5.3) can be rewritten [3] as :

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta}, \mathbf{s})}{\partial \boldsymbol{\theta}} = \sum_{j=1}^{T} \left(\sum_{i=1}^{N} \mathbf{D}_{i,j} \frac{\partial \mathcal{L}(\boldsymbol{\theta}, \mathbf{s})}{\partial G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)} \right) \frac{\partial g_{\boldsymbol{\theta}}(w^j, \mathbf{c}^j)}{\partial \boldsymbol{\theta}} \tag{5.4}$$

where $(w^j, \mathbf{c}^j)$ denotes the j^{th} n-gram. This gradient differs from the CLL gradient by weighting coefficients :

$$k_j = \sum_{i=1}^{N} \mathbf{D}_{i,j} \frac{\partial \mathcal{L}(\boldsymbol{\theta}, \mathbf{s})}{\partial G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)}$$

corresponding to $(w^j, \mathbf{c}^j)$. In practice, our gradient calculation follows closely this formulation. [4] We first compute the gradient of the loss function with respect to each hypothesis

[3] The second term is ignored for simplicity.

[4] The only difference is that gradient is computed not for each source sentence $\mathbf{s}$, but for a mini-batch of 128 sentences at once.

score, each of them is attributed to all n-grams extracted from this hypothesis. Then, we perform an *n-gram grouping* process in which coefficients of the same n-gram from different hypotheses, or different places in the same hypothesis, are summed up. It also helps to reduce the number of n-grams to be handled, and to speed up the training. Finally, $\boldsymbol{\theta}$ is updated using the gradients of n-gram-level scores (with respect to $\boldsymbol{\theta}$), multiplied by the corresponding coefficient k_j resulted from the n-gram grouping step. Following this procedure, the introduction of new forms for the criterion $\mathcal{L}(\boldsymbol{\theta}, \mathbf{s})$ involves changes only in the first term of (5.3).

5.2 Discriminative training criteria for CSMs

The generic Algorithm 5 can incorporate any objective function $\mathcal{L}(\boldsymbol{\theta}, \mathbf{s})$. However, the discriminative framework exhibits its full advantage compared to the MLE only if $\mathcal{L}(\boldsymbol{\theta}, \mathbf{s})$ aims at optimizing the performance of the whole system, and if the optimization is guided by MT metrics (such as BLEU), instead of by the perplexity of the CSTM. This section aims to explore several such objective functions.

5.2.1 A max-margin approach

Our max-margin training criterion is derived from the structured perceptron approach (Collins, 2002; Taskar et al., 2004) described in Section 1.3.1. As explained above, each hypothesis $\mathbf{h}_i$ produced by the decoder is scored according to (5.2). Its quality can also be evaluated by a sentence-level approximation of the BLEU score $SBLEU(\mathbf{h}_i)$ (Nakov et al., 2012). Let $\mathbf{h}^*$ be the hypothesis with the best sentence-BLEU score in $\mathbf{H_s}$. A max-margin training criterion (Freund and Schapire, 1999; McDonald et al., 2005; Watanabe et al., 2007) can then be formulated as follows :

$$\mathcal{L}_{mm}(\boldsymbol{\theta}, \mathbf{s}) = \max_{1 \leq j \leq N} \left[\text{cost}_\alpha(\mathbf{h}_j) + G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_j) - G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}^*) \right] \tag{5.5}$$

where the cost function, defined as :

$$\text{cost}_\alpha(\mathbf{h}_j) = \alpha(SBLEU(\mathbf{h}^*) - SBLEU(\mathbf{h}_j)) \tag{5.6}$$

reflects the cost paid by choosing $\mathbf{h}_j$ instead of the best hypothesis $\mathbf{h}^*$. This objective function is similar to the loss (1.12) used to tune $\boldsymbol{\lambda}$, except that here the general scoring function $G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(.)$ depends both on the neural network and the log-linear coefficients. We also multiply the sentence-BLEU difference by a factor α in order to adjust the margin according to the scale of system-level scores. Taking $\alpha = 0$ transforms the general max-margin criterion to a structured perceptron loss (Collins, 2002). The goal is to discriminatively learn to give the highest model score to the hypothesis $\mathbf{h}^*$. Moreover, the margin term enforces the scoring difference between $\mathbf{h}^*$ and the rest of the N-best list to be greater than the BLEU difference.

To compute the gradient for training purpose, we use the following gradient :

$$\frac{\partial \mathcal{L}_{mm}(\boldsymbol{\theta}, \mathbf{s})}{\partial G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)} = \begin{cases} -1, & \text{if } \mathbf{h}_i = \mathbf{h}^* \\ 1, & \text{if } \mathbf{h}_i = \underset{1 \leq j \leq N}{\operatorname{argmax}} \left[\text{cost}_\alpha(\mathbf{h}_j) + G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_j)\right] \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

which assigns non-zero terms only to the oracle hypothesis $\mathbf{h}^*$ and another one obtained from the *argmax* operation. The former derivation can be identified before the training starts (as N-best lists are kept fixed during training), while the latter only requires to be chosen from the N-best list. The framework of N-best list rescoring henceforth helps us to simplify the gradient computation for the update of $\boldsymbol{\theta}$.

However, a source sentence $\mathbf{s}$ can have, within the N-best list, several good translations that differ only slightly from the best hypothesis. The max-margin objective function defined above nevertheless considers that all hypotheses, except the best one, are wrong. The ranking-based approach defined below tries to correct this weakness.

5.2.2 Pairwise ranking

Inspired by the pairwise ranking approach (Hopkins and May, 2011), we define another criterion that aims to simulate the model score to follow the ranking of hypotheses according to their sentence-BLEU scores. Let r_i be the rank of the hypothesis $\mathbf{h}_i$ [5] when the N-best list is reordered by the sentence-level BLEU, this criterion is defined as :

$$\mathcal{L}_{pro}(\boldsymbol{\theta}, \mathbf{s}) = \sum_{1 \leq i,k \leq N} \mathbb{I}_{\{r_i + \delta \leq r_k, G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s},\mathbf{h}_i) < G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s},\mathbf{h}_k)\}} \left(-G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i) + G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_k)\right). \tag{5.8}$$

Note that this loss function does not involve all hypothesis pairs, since two hypotheses are included in the sum only if they are sufficiently apart in terms of their ranks. Formally, the absolute difference of ranks should be greater than a threshold δ, here δ is an predefined integer hyper-parameter. The ranking problem is thus reduced to a binary classification task taking candidate translation pairs as inputs.

However, a major weakness of this kind of criteria is that not all candidate pairs are equally relevant to give a high BLEU score. Instead, a heavily mis-ranked hypothesis pair, in which the best hypothesis is much better than the other in terms of BLEU, but is undermined by the system, should be taken into account with a strong emphasis. PRO described in (Hopkins and May, 2011) resolves this problem by resorting to a sampling routine by which hypothesis pairs are sampled based on the BLEU difference between the two hypotheses inside each pair. The procedure ensures that a more relevant pair (with large difference in terms of sentence-BLEU score) will be included into the loss function more frequently than a less relevant one. In this work, we however advocate an adaptation of margins : instead of adopting a margin fixed to 1 as in Equation (1.13), we use a cost-based margin as defined in Equation (5.5). More precisely, the ranking criterion (5.8) is generalized in such a way that for each pair of hypotheses $(\mathbf{h}_i, \mathbf{h}_k)$ such as $r_i + \delta < r_k$, the scoring difference $G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i) - G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_k)$ should exceed a positive

[5] The best hypothesis has rank 1, while the worst has rank equal to the size of the N-best list.

margin which is $\alpha(SBLEU(\mathbf{h}_i) - SBLEU(\mathbf{h}_k))$. Let us define the set of all critical pairs of hypotheses :

$$\mathcal{C}^{\alpha}_{\delta,pro-mm} = \{(\mathbf{h}_i, \mathbf{h}_k) : 1 \leq i,k \leq N, r_i + \delta \leq r_k, G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i) - G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_k) < \text{cost}_{\alpha}(\mathbf{h}_k) - \text{cost}_{\alpha}(\mathbf{h}_i)\} \tag{5.9}$$

where the cost function $\text{cost}_{\alpha}(.)$ is still defined by (5.6). Then the objective function that combines both the pairwise ranking and max-margin criteria is defined as follows :

$$\mathcal{L}_{pro-mm}(\boldsymbol{\theta}, \mathbf{s}) = \sum_{(\mathbf{h}_i,\mathbf{h}_k)\in\mathcal{C}^{\alpha}_{\delta}} \text{cost}_{\alpha}(\mathbf{h}_k) - \text{cost}_{\alpha}(\mathbf{h}_i) - G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i) + G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_k). \tag{5.10}$$

Taking $\alpha = 0$, this function is equivalent to the pairwise ranking training (5.8). $\alpha > 0$ modifies the set of critical pairs : at first glance it appears only to extend this set, as $\text{cost}_{\alpha}(\mathbf{h}_k) - \text{cost}_{\alpha}(\mathbf{h}_i) \geq 0$. However it also discriminates relevant pairs from the full set of all pairs. Indeed, the larger the BLEU difference is, the harder it is to remove the corresponding pair $(\mathbf{h}_i, \mathbf{h}_k)$ from $\mathcal{C}^{\alpha}_{\delta}$. In other words, more relevant pairs are used by the optimization a greater number of times than less relevant ones. As a consequence, this definition of margins yields a similar effect without explicitly introducing the sampling routine employed in PRO. Moreover, we can flexibly adjust its contribution throughout the hyper-parameter α.

Finally, all discriminative criteria that have been described can be rewritten under the following form :

$$\mathcal{L}(\boldsymbol{\theta}, \mathbf{s}) = \sum_{(\mathbf{h}_i,\mathbf{h}_k)\in\mathcal{C}^{\alpha}_{\delta}} \text{cost}_{\alpha}(\mathbf{h}_k) - \text{cost}_{\alpha}(\mathbf{h}_i) - G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i) + G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_k) \tag{5.11}$$

with the critical set $\mathcal{C}^{\alpha}_{\delta}$ being defined by each approach. For the max-margin criterion (5.5), $\mathcal{C}^{\alpha}_{\delta}$ contains only one hypothesis pair :

$$\mathcal{C}^{\alpha}_{\delta,mm} = \left\{(\mathbf{h}^{*\mathbf{s}}, \mathbf{h}_i), \mathbf{h}_i = \underset{1\leq j\leq N}{\text{argmax}} \left(\text{cost}_{\alpha}(\mathbf{h}_i) + G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)\right)\right\} \tag{5.12}$$

where the cost function of $\mathbf{h}^{*\mathbf{s}}$ is zero : $\text{cost}_{\alpha}(\mathbf{h}^{*\mathbf{s}}) = 0$. For the pairwise ranking criterion, the set has already been defined by (5.9). A variant of this criterion defines the threshold δ in terms of sentence-BLEU score as does the PRO tuning algorithm (Hopkins and May, 2011), and considers the following critical set :

$$\begin{aligned}\mathcal{C}^{\alpha}_{\delta,pro-mm-sbleu} =& \{(i,k) : SBLEU(\mathbf{h}_k) + \delta \leq SBLEU(\mathbf{h}_i), \\ & G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i) - G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_k) < \text{cost}_{\alpha}(\mathbf{h}_k) - \text{cost}_{\alpha}(\mathbf{h}_i)\}\end{aligned} \tag{5.13}$$

which specifies a variant of our PRO-MM criterion. For the rank-based threshold, δ is set to 250 for 300-best lists, while the sentence-BLEU based threshold is set to 0.05.

No matter how this set needs to be defined, the gradient with respect to the score $G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)$ will be the number of times $\mathbf{h}_i$ appears in $\mathcal{C}^{\alpha}_{\delta}$ at the second position in a pair (*negative* example), subtracted by the number of times it appears at the first position (*positive* example).

5.2.3 Maximizing conditional log-likelihood on N-best lists

We have so far considered the hypothesis model scores in terms of their absolute values. These quantities can also be probabilistically interpreted, by assigning a probability to each hypothesis in the N-best list $\mathbf{H_s}$ as follows :

$$\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{h}_i|\mathbf{s}) = \frac{\exp(\beta \times G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s},\mathbf{h}_i))}{\sum_{\mathbf{h}\in\mathbf{H_s}} \exp(\beta \times G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s},\mathbf{h}))} \tag{5.14}$$

which is similar to Equation (1.11) discussed in Chapter 1, except that here the log-linear scoring function is also considered as a function of $\boldsymbol{\theta}$. Thanks to this interpretation, we can define an N-best-list-based CLL criterion which is the negative log-likelihood of the oracle translation $\mathbf{h}^{*\mathbf{s}}$ given the source sentence $\mathbf{s}$ and the corresponding $\mathbf{H_s}$:

$$\begin{aligned}\mathcal{L}_{cll,N-best}(\boldsymbol{\theta},\mathbf{s}) &= -\log \mathbf{p}_{\boldsymbol{\theta}}(\mathbf{h}^{*\mathbf{s}}|\mathbf{s}) \\ &= -\beta \times G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s},\mathbf{h}^{*\mathbf{s}}) + \log\left(\sum_{\mathbf{h}\in\mathbf{H_s}} \exp(\beta \times G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s},\mathbf{h}))\right)\end{aligned} \tag{5.15}$$

where $\mathbf{h}^{*\mathbf{s}}$ defines the best BLEU hypothesis in the N-best list.

The gradient term corresponding to this objective function is computed as follows : [6]

$$\frac{\partial \mathcal{L}_{cll,N-best}(\boldsymbol{\theta},\mathbf{s})}{\partial G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s},\mathbf{h}_i)} = -\mathbb{I}_{\mathbf{h}_i=\mathbf{h}^{*\mathbf{s}}} + \mathbf{p}_{\boldsymbol{\theta}}(\mathbf{h}_i|\mathbf{s})$$

where $\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{h}_i|\mathbf{s})$ is computed by (5.14). The general gradient computation has already been described in Section 5.1.3.

5.2.4 Expected-BLEU and ListNet optimizations

We also derive the expected-BLEU and ListNet optimizations for the CSTM from the corresponding objective functions used to train the log-linear coefficients (Equation (1.10) and (1.16)). Here we rewrite these functions under the following forms :

$$\mathcal{L}_{expected-\mathrm{BLEU}}(\boldsymbol{\theta},\mathbf{s}) = -\mathrm{xBLEU}^{\mathbf{s}}_{\boldsymbol{\theta}} = -\sum_{\mathbf{h}_i\in\mathbf{H_s}} SBLEU(\mathbf{h}_i)\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{h}_i|\mathbf{s}) \tag{5.16}$$

$$\mathcal{L}_{ListNet}(\boldsymbol{\theta},\mathbf{s}) = -\sum_{\mathbf{h}_i\in\mathbf{H_s}} \mathbf{p}_{SBLEU}(\mathbf{h}_i) \times \log \mathbf{p}_{\boldsymbol{\theta}}(\mathbf{h}_i|\mathbf{s}) \tag{5.17}$$

where $\mathrm{xBLEU}^{\mathbf{s}}_{\boldsymbol{\theta}}$ denotes the expected sentence-BLEU score computed on hypotheses in $\mathbf{H_s}$, and according to the probability distribution $\mathbf{p}_{\boldsymbol{\theta}}(.)$ (Equation (5.14)), while $\mathbf{p}_{SBLEU}(.)$ is another distribution on $\mathbf{H_s}$ derived from sentence-BLEU scores computed by :

$$\mathbf{p}_{SBLEU}(\mathbf{h}_i) = \frac{\exp(\gamma \times SBLEU(\mathbf{h}_i))}{\sum_{\mathbf{h}\in\mathbf{H_s}} \exp(\gamma \times SBLEU(\mathbf{h}))}$$

[6] β appears in the corresponding gradient, just as in case of Expected-BLEU and ListNet optimizations, but we ignore it in the formula as well as in our implementation as it only implies a change in learning rates.

Config.	Domain / dev/test	SMT system (1)	Initialization CSM	Train data CSM (2)	Relationship (1) & (2)
random init.	TED'2011/ tst2010/ dev2010	WMT'13 n-code (12M)	random	107,058 of TED	$(2) \cap (1) = \emptyset$ $\lvert(2)\rvert \ll \lvert(1)\rvert$
WMT CSTM adapted			from CLL CSTM on WMT		
Mono-init TED (a)			from NNLMs on TED		
CLL (a) fine-tuned			from CLL model of (a)		
Mono-init WMT (b)			from NNLMs on WMT		
CLL (b) fine-tuned			from CLL model of (b)		
Training TED 2014	TED'2014/ ted.dev/ ted.test	n-code on TED (180K)	from CLL model on TED	180K of TED	$(2) = (1)$
Adaptation TED 2014		n-code on WMT (12M)			$(2) \cap (1) = \emptyset$ $\lvert(2)\rvert \ll \lvert(1)\rvert$
Partial training on medical	Medical/ devel/ test	n-code on medical **incl.** P-A	from CLL model on Patent-Abstract	Patent-Abstract (200K)	$(2) \subset (1)$ $\lvert(2)\rvert \ll \lvert(1)\rvert$
"Adaptation" on Medical		n-code on medical **except** P-A			$(2) \cap (1) = \emptyset$ $\lvert(2)\rvert \ll \lvert(1)\rvert$

Table 5.1 – Details about the experimental set-ups.

in which, following (Niehues et al., 2015), we set γ to 100. The gradient terms of these two criteria are :

$$\frac{\partial \mathcal{L}_{expected-\mathrm{BLEU}}(\boldsymbol{\theta}, \mathbf{s})}{\partial G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)} = \mathbf{p}_{\boldsymbol{\theta}}(\mathbf{h}_i|\mathbf{s}) \times (\mathrm{xBLEU}^{\mathbf{s}}_{\boldsymbol{\theta}} - SBLEU(\mathbf{h}_i))$$

$$\frac{\partial \mathcal{L}_{ListNet}(\boldsymbol{\theta}, \mathbf{s})}{\partial G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}_i)} = \mathbf{p}_{\boldsymbol{\theta}}(\mathbf{h}_i|\mathbf{s}) - \mathbf{p}_{SBLEU}(\mathbf{h}_i)$$

5.3 Experimental configurations

In order to assess the impact of different strategies on the discriminative framework, we perform experiments on training and adaptation tasks in two different domains. The first domain is derived from a lecture translation task (TED Talks task), whereas the second domain is the medical translation task of WMT'2014.

5.3.1 Task and corpora

The first set of experiments is carried out in a Domain Adaptation situation, where in-domain data corresponds to a lecture translation task, and the baseline system is a state-of-the-art out-of-domain SMT system which has been intensively trained for a news translation task. The goal is therefore to quickly and efficiently adapt the SMT system by only adapting the CSTM. The task considered here is derived from the text translation track of IWSLT'2011 from English to French (the TED Talks task (Federico et al., 2012)), where a (in-domain) training dataset containing $107,058$ aligned sentence pairs is made available. As explained above, this corpus serves only to adapt the CSTM, i.e to adapt the parameters $\boldsymbol{\theta}$. The baseline and out-of-domain system is trained in the condition of the shared translation task of WMT'2013 evaluation campaign. [7] This system includes a CSTM that will be used as the starting point for adaptation. The official development and test sets respectively contain 934 and $1,664$ sentence pairs. Following (Le et al., 2012a), these sets are swapped, the tuning of the log-linear coefficients $\boldsymbol{\lambda}$ is carried out on $1,664$ sentences of the latter, while the final test is on 934 sentences of the former.

To investigate the impact of model initialization on the performance of the discriminative framework, the experimental configuration is also extended to include other initialization schemas for CSTMs (see the upper part of Table 5.1). For SOUL CSTMs, the model can be initialized using monolingual NNLMs trained on in-domain (*mono-init TED (a)*) or out-of-domain (*mono-init WMT (b)*) corpora. The NNLMs serve as a tool to build the output vocabulary hierarchical structure; this structure is then fixed for further training and building of the initial bilingual models (Section 3.1.2). For NCE models, we initialize $\boldsymbol{\theta}$ using in-domain NNLMs. These initialization scenarios are compared with the random initialization, and with the scenario in which discriminative training is only considered as a fine-tuning phase for CSTMs which have been pre-trained by the CLL criterion on TED data (*CLL (a) & (b) fine-tuned*).

For the second set of experiments, the discriminative framework is evaluated both in training and adaptation scenarios (the lower part of Table 5.1). In the *training* scenario, the CSTM is trained on the same parallel data as the one used to train the baseline SMT system. In the *adaptation* scenario, as described above, large out-of-domain corpora are used to train the baseline SMT system, while the CSTM is trained on a much smaller, in-domain corpus. An intermediate situation (*partial training*) is when only a fraction of the SMT training data is reused to estimate the CSTM : this situation is interesting because it allows us to train the CSTM much faster than in the training scenario.

Two domains are investigated in this set of experiments. For the TED Talks task, we use the 2014 version [8] which contains $180K$ in-domain sentence pairs. The out-of-domain is still the corpora accepted for the English-to-French shared translation task of WMT'2013. The second domain is the medical translation task of WMT'2014 [9] (English-to-French) for which we use all authorized in-domain corpora (above 4M sentence pairs). The Patent-Abstract corpus, made of 200K sentence pairs, is used either for CSTM adaptation or partial training. Note that a similar set-up has been described in Chapter 3

[7] http://www.statmt.org/wmt13/ .
[8] http://workshop2014.iwslt.org/ .
[9] www.statmt.org/wmt14/medical-task/ .

(Table 3.1), except that here we discriminatively train or adapt the CSTMs; moreover, the SMT system varies (the lower part of Table 5.1) in order to examine its impact on the performance of the discriminative framework (Section 5.4.4). We pay particular attention to the relationship between the dataset used to train CSTMs and the one used for SMT system training; this relationship decides whether the current task is of *adaptation* or *training* situation. The relative sizes of corpora are also worth some attention.

Finally, the *Adaptation TED* 2014 (Table 5.1) is rerun with several discriminative criteria in order to assess the impact of these criteria on the translation performance of the final system (Section 5.4.3). The discriminative framework is proved to be very efficient in this configuration; the translation performance however depends strongly on the objective function used to update the model parameters $\boldsymbol{\theta}$. In all experiments described in this chapter, the batch N-best MIRA, as described in Section 1.3.1 and implemented in MOSES, is used to tune $\boldsymbol{\lambda}$.

5.3.2 Translation system and model structure

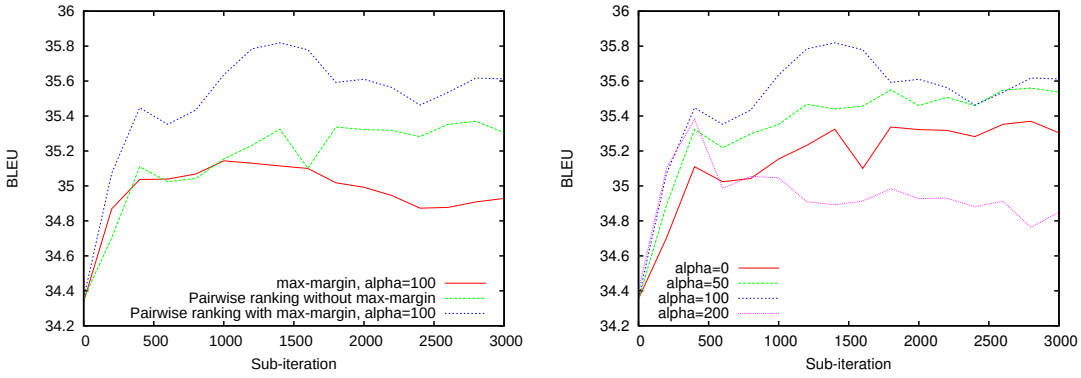
The SMT systems are based on the bilingual n-gram approach to SMT described in Section 1.1.3 and constructed from an open source implementation. [10] The CSTM structure is the lexicalized model based on the n-gram approach (Section 2.4.2) for which we still use hyper-parameters summarized in Table 3.2. As formulated in Section 2.4.2, four neural network models can be learnt which correspond to various factorizations of $\mathbf{p}(\mathbf{t}, \mathbf{s}|\mathbf{a})$:

$$\begin{aligned}\mathbf{p}(\mathbf{t}, \mathbf{s}|\mathbf{a}) &= \prod_{l=1}^{L} \left[\prod_{k=1}^{|\bar{t}_l|} \mathbf{p}(t_l^k|\mathbf{c}_{n-1}(t_l^k), \mathbf{c}_{n-1}(s_{l+1}^1)) \times \prod_{k=1}^{|\bar{s}_l|} \mathbf{p}(s_l^k|\mathbf{c}_{n-1}(t_l^1), \mathbf{c}_{n-1}(s_l^k)) \right] \\ &= \prod_{l=1}^{L} \left[\prod_{k=1}^{|\bar{s}_l|} \mathbf{p}(s_l^k|\mathbf{c}_{n-1}(s_l^k), \mathbf{c}_{n-1}(t_{l+1}^1)) \times \prod_{k=1}^{|\bar{t}_l|} \mathbf{p}(t_l^k|\mathbf{c}_{n-1}(s_l^1), \mathbf{c}_{n-1}(t_l^k)) \right]\end{aligned}$$

For the sake of clarity, we focus our study on models estimating $\mathbf{p}(t_l^k|\mathbf{c}_{n-1}(t_l^k), \mathbf{c}_{n-1}(s_{l+1}^1))$ and $\mathbf{p}(t_l^k|\mathbf{c}_{n-1}(s_l^1), \mathbf{c}_{n-1}(t_l^k))$. Similar trends were observed with other CSTMs.

For the discriminative training and adaptation tasks, baseline SMT systems are used to generate respectively 600 and 300 best hypotheses for each sentence in the in-domain corpus. The threshold δ in the PRO-MM criterion (Equation (5.9)) is set to 250 for 300-best, and to 500 for 600-best lists. The sentence-BLEU-based threshold is set to 0.05 (Equation (5.13)), while α is set empirically. As a convention throughout all experiments in this dissertation, reported BLEU scores are averaged over 8 MIRA runs from random initial points. For a fair comparison, all BLEU scores reported are obtained after a tuning phase on the development set, including out-of-domain systems.

[10] perso.limsi.fr/Individu/jmcrego/bincoder.



(a) (b)

Figure 5.2 – (a) : Evolution of BLEU scores on the dev set using three discriminative criteria described in (5.5), (5.8) and (5.10). Vector $\boldsymbol{\lambda}$ is updated every 200 sub-iterations (mini-batches). (b) : Evolution of BLEU scores on the dev set with different values of α. $\mathcal{L}_{pro-mm}$ is used in all cases.

5.4 Experimental results

5.4.1 Domain Adaptation on TED Talks

We first consider the Domain Adaptation task as described in Section 5.1.2. The experiments aim at comparing the discriminative adaptation framework (Figure 5.1) with the usual method (Le et al., 2012a) where CSTMs are retrained with the new in-domain data. The experimental configuration is *WMT CSTM adapted* from Table 5.1 and only SOUL CSTMs are used. Figure 5.2a compares the three discriminative criteria respectively defined in (5.5), (5.8) and (5.10) in terms of BLEU scores on the development set when adapting the model estimating $\mathbf{p}(t_l^k|\mathbf{c}_{n-1}(t_l^k), \mathbf{c}_{n-1}(s_{l+1}^1))$. According to these results, the pairwise ranking criterion, with or without max-margin, clearly outperforms the max-margin approach (5.5). This result outlines the benefit of using criteria based on multiple hypotheses from different parts of the N-best list, rather than only on the pair of *hope* and *fear* candidates as does the max-margin loss. Indeed, the gradient of the max-margin criterion computed by (5.7) gives non-zero terms only to two hypotheses, while the pairwise ranking objective involves much more candidates in the optimization of $\boldsymbol{\theta}$. The participation of a large number of hypotheses would give to the pairwise ranking training more reliable information about which word choices need to be rewarded or penalized. Moreover, the pairwise ranking strategy also allows us to deal with multiple good translations situated in the upper part of the N-best list, instead of updating $\boldsymbol{\theta}$ solely toward the oracle translation.

To assess the impact of the margin in $\mathcal{L}_{pro-mm}$, we plot on Figure 5.2b the evolution of the BLEU score on the development set as a function of α. When $\alpha = 0$, the objective function, equivalent to the pairwise ranking $\mathcal{L}_{pro}$, equally considers all potential pairs of hypotheses for which the difference in ranks is superior to δ. By increasing α, the optimization focusses more strongly on pairs with large difference in terms of sentence-BLEU score. This discrimination corresponds to an improvement of 0.4 BLEU point in

System	*dev*	*test*
Baseline systems (out-of-domain)		
n-code	33.9	27.6
n-code + CSTM WMT	34.4	28.5
Adapted systems		
n-code + CSTM CLL adapted	34.9	29.0
n-code + CSTM $\mathcal{L}_{mm}$ adapted $\alpha = 100$	35.2	29.3
n-code + CSTM $\mathcal{L}_{pro}$ adapted	35.4	29.4
n-code + CSTM $\mathcal{L}_{pro-mm}$ adapted $\alpha = 100$	**35.8**	**29.6**

Table 5.2 – BLEU scores obtained for different adaptation schemes of the CSTM for $\mathbf{p}(t_l^k|\mathbf{c}_{n-1}(t_l^k), \mathbf{c}_{n-1}(s_{l+1}^1))$ with WMT baselines : maximum conditional likelihood (CLL) *vs* discriminative adaptation. The log-linear coefficients of the baseline systems are re-tuned using the in-domain dev set.

our experiments, while beyond $\alpha = 100$, the performance starts to drop.

The results of adapting the model estimating $\mathbf{p}(t_l^k|\mathbf{c}_{n-1}(t_l^k), \mathbf{c}_{n-1}(s_{l+1}^1))$ are presented in Table 5.2. The upper part reports the baseline BLEU scores. Initial results are obtained with the out-of-domain one-pass system, and a 0.9 BLEU point improvement is obtained when rescoring its output with the out-of-domain CSTM. The lower part of Table 5.2 summarizes the results obtained with various adaptation methods : the conditional likelihood (CLL) optimization yields an additional increase of 0.5 BLEU point, which is doubled when using the discriminative objective function $\mathcal{L}_{pro-mm}$ to perform adaptation. As showed in the middle part of Table 5.3, similar improvements are obtained with the adaptation of the model estimating $\mathbf{p}(t_l^k|\mathbf{c}_{n-1}(s_l^1), \mathbf{c}_{n-1}(t_l^k))$.

Finally, the lower part of Table 5.3 compares the performance obtained by our discriminative adaptation method to the one published in (Le et al., 2012a) for a similar experimental set-up. In our experiment (the last line), in-domain data is only used to re-tune $\boldsymbol{\lambda}$ in the baseline system, and to perform discriminative adaptation for two CSTMs. In (Le et al., 2012a), the SMT system is entirely re-trained from scratch to integrate in-domain data (from word alignments to large scale target language model), and all four CSTMs defined by the last formulation in Section 2.4.2 are adapted using the CLL criterion. This experiment shows that we can achieve slightly better performance by only adapting two CSTMs with the proposed adaptation method.

5.4.2 Initialization issues

Initialization of the CSM parameters $\boldsymbol{\theta}$ and of the log-linear coefficients $\boldsymbol{\lambda}$ is an important issue when using the discriminative procedure. In (Gao et al., 2014; Auli and Gao, 2014), the authors initialize λ_{M+1} to 1, and normalize all other log-linear coefficients. In this dissertation, we however initialize λ_{M+1} by optimizing it on development set using initial parameter values $\boldsymbol{\theta}$ of the CSTM, as well as parameters of other component models which have been included in the SMT system. The initial values of $\boldsymbol{\theta}$ hence play an important role in Algorithm 5, as they also influence the initialization of $\boldsymbol{\lambda}$ which,

System	*dev*	*test*
Baseline systems (out-of-domain)		
n-code	33.9	27.6
n-code + CSTM WMT	34.6	28.2
Adapted systems		
n-code + CSTM CLL adapted	35.1	28.7
n-code + CSTM $\mathcal{L}_{pro-mm}$ adapted $\alpha = 100$	**35.5**	**29.3**
Model combination		
n-code (+TED) + all CSTMs CLL adapted (Le et al., 2012a)	36	29.7
n-code + all WMT CSTMs + 2 CSTMs $\mathcal{L}_{pro-mm}$	**36.3**	**29.8**

Table 5.3 – BLEU scores obtained for different adaptation schemes of the CSTM for $\mathbf{p}(t_l^k|\mathbf{c}_{n-1}(s_l^1), \mathbf{c}_{n-1}(t_l^k))$ in the middle part, and with model combination in the lower part. The notation n-code (+TED) emphasizes that for this system the baseline SMT system is *re-trained* with out-of-domain and in-domain data, while in all other cases the baseline system only uses out-of-domain data.

	dev	**test**
Baseline system	33.9	27.6
Adding a standard CSTM		
NCE mono-init TED (a)	34.8	28.8
Adding a discriminatively trained CSTM		
random initialization	34.3	28.5
mono-init TED	35.2	29.1
CLL (a) fine-tuned	**35.4**	**29.7**
Oracle	46.1	39.0

Table 5.4 – Comparison of results obtained in terms of BLEU scores with different unnormalized CSTMs. The term *mono-init* (TED or WMT) indicates a CSTM initialized with monolingual models trained on the corresponding monolingual dataset.

altogether constitutes the starting point of our iterative procedure. In this section, we present an empirical study to evaluate different initialization scenarios for $\boldsymbol{\theta}$. These parameters can be randomly initialized, or get their values based on monolingual models as described in Section 3.1.2 (*mono-init TED / WMT*). The third scenario consists of using the discriminative procedure only in a fine-tuning phase, while the CSTM is pre-trained by optimizing its CLL (or an approximation using the NCE algorithm). This technique hence takes into account a mixture of information from various sources : combination between the n-gram-level information on word frequencies and the sentence-level information on translation hypotheses, combination between unsupervised (with SOUL or NCE) and supervised training guided by sentence-BLEU scores.

The TED Talks adaptation set-up is still used, except that various initialization scenarios are explored (Table 5.1), while both SOUL and NCE CSTMs are involved. The upper part of Table 5.1 summarizes several experimental configurations used in this section. The PRO-MM criterion (Equation (5.10)), which has obtained the best performance in

	dev	test
Baseline system	33.9	27.6
Adding a standard CSTM		
SOUL mono-init TED (a)	35.1	28.8
SOUL mono-init WMT (b)	35.2	28.9
Adding a discriminatively trained CSTM		
random initialization	33.8	27.6
mono-init TED	35.0	28.9
CLL (a) fine-tuned	35.7	29.3
mono-init WMT	35.2	**29.5**
CLL (b) fine-tuned	**35.9**	**29.5**
Oracle	46.1	39.0

Table 5.5 – Comparison of results obtained in terms of BLEU scores with SOUL-structure models.

the precedent experiment is employed and parameter α is set to 100. In order to compare different training methods and initializations, the first experiments involve un-normalized models, which can be trained using the NCE algorithm, the discriminative method, or a combination of these two.

Table 5.4 presents the results obtained with different training criteria and initialization techniques. The conventional CSTM structure, trained with NCE algorithm, gives an improvement of 1.2 BLEU points over the baseline system.

The second part of the table compares different results using the discriminative criterion (PRO-MM) and shows the importance of initialization for this training method. Starting from a random initialization, the discriminative model obtains a significant gain of 0.9 BLEU points, which however is less than the gain obtained by NCE CSTM. However, if we use the same initialization technique as with the NCE CSTM (i.e *mono-init TED* setting), the discriminative model yields an additional gain of 0.3 BLEU points compared to the NCE CSTM. Finally, the best result is obtained with the combination of these two criteria : the model is first pre-trained with the NCE algorithm, then fine-tuned using the discriminative framework. This configuration gives to the SMT system (with only one CSTM included) an impressive gain of 2.1 BLEU points, and this only with a simple integration using N-best rescoring.

Further experiments are carried out to perform the same comparison on the SOUL structure, with results presented in Table 5.5. We can observe that training SOUL models from initializations either on in-domain or out-of-domain data obtains similar results (only 0.1 BLEU point difference), and also similar to the one obtained with the NCE model. However, the performance of the discriminative framework strongly depends on the initialization value. Indeed, starting from a random initialization, we do not obtain any improvement compared to the baseline system. When using the initialization strategy based on NNLMs (which is also employed by SOUL models trained with the CLL criterion), the performance depends on whether the NNLMs have been trained on in-domain (*mono-init TED* setting) or out-of-domain (*mono-init WMT* setting) corpora. Our experimental results show that initializing using out-of-domain data significantly outperforms

the one using in-domain data, no doubt because of the effect of combining these two corpora in the first scenario. More interesting is the fact that, if only in-domain data is used, the discriminative method does not seem to outperform the CLL optimization from the same *mono-init TED* starting point (28.9 versus 28.8); however when out-of-domain data is used (the *mono-init WMT* setting), the discriminative method clearly outperforms the CLL training (29.5 versus 28.9). It seems likely that discriminatively trained models are capable of remembering information provided during the initialization step, while CLL-trained models fail to exploit it. This capacity could be attributed to the word embeddings which are kept fixed during our discriminative adaptation. [11] This feature becomes relevant in situations where data comes from different sources and domains, such as Domain Adaptation tasks described in the previous experiment.

In all experiments with SOUL and NCE CSTMs, the combination of n-gram-level and sentence-level training criteria (the *fine-tuning* configurations) obtains the best performance. This observation is related to the work of (Collobert and Weston, 2008; Collobert et al., 2011) in which the authors use some unlabelled training data to learn word embeddings, before using these feature vectors for supervised tasks. Compared to the expected-BLEU training, described for instance in (Auli and Gao, 2014), our method has an advantage that comes from n-gram-level information and pre-trained word feature vectors.

All in all, starting the discriminative procedure on top of an NCE model seems to deliver the best result. Moreover, NCE trained models are un-normalized, which reduces the computational cost of computing the scores of N-best lists for the entire CSTM training set. This is hence our preferred experimental configuration for the rest of this chapter.

5.4.3 A comparison between discriminative criteria

We extend the comparison between max-margin and pairwise-ranking approaches (Section 5.4.1) to also include other discriminative criteria described in Section 5.2. The experiments are conducted in the *Adaptation TED* 2014 configuration of Table 5.1, where the conventional development and test sets are used for tuning $\boldsymbol{\lambda}$ and for testing. Following the conclusion from the precedent section, we pre-train the CSTM using NCE algorithm on the TED Talks corpus, and use it as the starting point for the discriminative procedure.

Table 5.6 compares the results obtained under this configuration using the different criteria. The upper part reports baseline scores on the development and test sets, while the second part presents improvements obtained by integrating the NCE-trained CSTM into the baseline system. Adding the CSTM into the SMT system yields gains of 0.7 BLEU points on the development set, and of up to 1.0 BLEU on the test set. Continuing the training process with the discriminative procedure amplifies these gains respectively to 1.3 and 2.1 BLEU in case the PRO-MM criterion is used.

The two versions of the threshold δ described in Section 5.2.2 here obtain similar results

[11] We have also experimented with conditions where the embeddings were modified; however no significant difference was observed.

Criterion	train	**dev**	**test**
	Baseline system		
	33.3	28.5	32.0
	Baseline + CSTM NCE		
	34.9	29.2	33.0
	Baseline + CSTM discriminative		
Max-margin $\alpha = 100$	34.9	29.6	32.9
PRO	35.3	29.6	33.4
PRO-MM $\alpha = 75$	**35.9** (+ **1.0**)	**29.8** (+ **0.6**)	**34.1** (+ **1.1**)
PRO SBLEU-threshod	35.2	29.5	33.5
PRO-MM SBLEU-threshold $\alpha = 100$	35.7	29.7	34.0
CLL on N-best list	35.5	**29.8** (+ **0.6**)	33.5
expected-BLEU	34.7	29.2	33.0
ListNet	35.4	29.7	33.5

Table 5.6 – Comparing the performance of NCE models trained using different discriminative criteria.

(34.1 versus 34.0). By comparing the 4 PRO-like criteria listed in Table 5.6, we realize the importance of margins within these criteria. As has been discussed in Section 5.2.2, setting $\alpha > 0$ allows the optimization to focus on relevant hypothesis pairs, as they are more likely to be used for the updates of $\boldsymbol{\theta}$. The use of margins in our PRO-MM criterion produces further gains of up to 0.7 BLEU points compared to the PRO criterion without margins ($\alpha = 0$). The failure of the max-margin criterion by itself to improve BLEU scores suggests however that the use of a critical set $\mathcal{C}_{\delta}^{\alpha}$ containing multiple hypothesis pairs (instead of using only one pair) is also crucial to obtain better results.

The two objective functions, CLL (on N-best lists) and ListNet, achieve reasonable gains that are equivalent to the results of the PRO without margins. It is important to notice that in these criteria, no margin term has been employed. A combination of the CLL and large-margin optimizations via a cost-augmented score has been described and investigated in (Gimpel and Smith, 2010) in the framework of linear structured prediction models.

The most surprising fact is that, contrarily to previous work (Gao et al., 2014; Auli and Gao, 2014), the expected-BLEU does not give any improvement in our experiments. In order to further investigate the behaviour of this criterion, we also compute BLEU scores on the training N-best lists [12] which have been used to discriminatively train the CSTM. The results in Table 5.6 show that there is a strong correlation between BLEU scores on the training and test sets. It is observed that the expected-BLEU criterion also fails to improve BLEU scores (with respect to the *Baseline + NCE CSTM* setting) even on the training data. This could be due to the fact that while the model parameters $\boldsymbol{\theta}$ are optimized using the expected-BLEU, the log-linear coefficients $\boldsymbol{\lambda}$ are tuned using N-best MIRA which corresponds to another criterion (the hinge loss, or max-margin, as described in Section 1.3.1). Figure 5.3, which plots a typical trend of the expected-BLEU score on the development set during training, illustrates this phenomenon. While this score im-

[12] Like scores on the development and test sets, these scores are also averaged over 8 MIRA runs from random initializations.

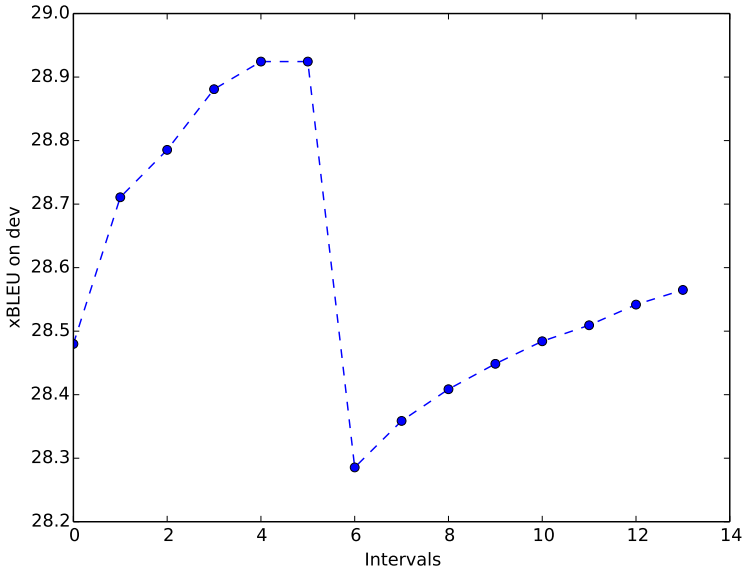


Figure 5.3 – Expected-BLEU scores on the development set re-evaluated at constant time intervals during training using the expected-BLEU criterion. The x-axis represents the training time intervals, while the y-axis represents the corresponding BLEU scores.

proves after each time interval, a re-optimization of $\boldsymbol{\lambda}$ [13] (using N-best MIRA on the same development set), brings this quantity down. It seems that an incompatibility between the training and tuning criteria is the main factor that explains the poor performance of this criterion. While only the batch N-best MIRA is tested in these experiments, the relative performance of criteria may change with other tuning methods, such as MERT, PRO, expected-BLEU (Zens et al., 2007) or a ListNet optimizer (Niehues et al., 2015). The results hence suggest the importance of using the same criterion for the training of CSTM parameters $\boldsymbol{\theta}$ and the tuning of log-linear coefficients $\boldsymbol{\lambda}$. We plan an extension of these experiments with other tuning algorithms to have a more complete conclusion and to investigate the importance of the consitency between these training criteria. Note that a related comparative study on the tuning task has been performed in (Zens et al., 2007).

Although using the same criterion with the tuning of $\boldsymbol{\lambda}$, the performance of the max-margin criterion presented in Table 5.6 suggests that always updating towards the fixed oracle candidate $\mathbf{h}^{*\mathbf{s}}$ is too risky. The PRO approach overcomes this problem, while an incorporation of margins makes it the best companion of the MIRA tuning algorithm in our training procedure.

5.4.4 The discriminative training of CSTMs

The experiments so far have been carried out in an adaptation framework in which the baseline SMT system is trained on out-of-domain data, while a small amount of in-domain data is used to train the CSTM. Despite its small size, this in-domain data is shown to be very effective to boost the performance of the baseline system. This experimental

[13] corresponding to the 6^{th} interval in the figure.

	train	dev	test
Training TED 2014			
Baseline n-code on TED	**65.6**	28.1	32.3
Baseline + CSTM NCE	64.1	28.9	33.1
Baseline + CSTM discriminative	64.9	**29.0** (+ 0.1)	**33.5** (+ 0.4)
Oracle	78.2	39.1	47.4
Adaptation TED 2014			
Baseline n-code on WMT	33.3	28.5	32.0
Baseline + CSTM NCE	34.9	29.2	33.0
Baseline + CSTM discriminative	**35.9**	**29.8** (+ 0.6)	**34.1** (+ 1.1)
Oracle	47.6	37.9	46.1

Table 5.7 – BLEU scores for the TED Talks tasks. In each task, the in-domain corpus is first used to train the CSTM using NCE, which constitutes the initialization for the discriminative procedure.

	train	dev	test
Partial training on medical			
Baseline n-code **incl.** Patent-Abstract	45.8	40.4	37.4
Baseline + CSTM NCE	45.2	40.8	38.1
Baseline + CSTM discriminative	**46.0**	**41.8** (+ 1.0)	**38.8** (+ 0.7)
Oracle	57.6	56.0	52.7
"Adaptation" on medical			
Baseline n-code **except** Patent-Abstract	39.4	39.8	37.2
Baseline + CSTM NCE	40.4	41.2	38.2
Baseline + CSTM discriminative	**41.5**	**41.8** (+ 0.6)	**38.9** (+ 0.7)
Oracle	50.7	55.6	52.5

Table 5.8 – BLEU scores for the medical tasks. The *Partial training* scenario uses the Patent-Abstract corpus to train the baseline SMT system, while the adaptation scenario does not. The CSTM is first trained using $200K$ sentences from the Patent-Abstract corpus, before being discriminatively trained on N-best lists generated on these same sentences.

condition is attractive as it provides evidence that, in order to adapt the SMT system to other domains, one might need only to retrain the CSTM. In this section, we further investigate the impact of the baseline SMT system over the CSTM discriminative training. Our experiments aim to identify the necessary conditions that need to be satisfied by the baseline system in order to guarantee a successful training process. The experimental configurations are summarized in the lower part of Table 5.1. From previously drawn conclusions, we only use the PRO-MM criterion with a rank-based threshold [14] δ set to 250 for 300-best, and to 500 for 600-best lists.

Results in Table 5.7 measure the impact of the discriminative training on top of an NCE model for the two TED Talks configurations. In the adaptation task, the discriminative training gives a large improvement of 1.1 BLEU points over the CSTM only trained

[14] Compared to the rank-based threshold, PRO-MM with a sentence-BLEU based threshold makes the training slower, however does not obtain better performance.

with NCE, and 2.1 BLEU points over the baseline SMT system. However, for the training scenario, these gains are reduced respectively to 0.4 and 1.2 BLEU points. The BLEU scores (in **train** column) measured on the training N-best lists gives insight to this difference : in training, these N-best lists contain hypotheses with an overoptimistic BLEU score, to be compared with the ones observed on unseen data. As a result, adding the CSTM significantly worsens the performance on the training data, contrarily to what is observed on the development and test sets. Even if the results of these two conditions cannot be directly compared (as the baselines are different), it seems that the proposed discriminative training has a greater impact on performance in the adaptation scenario, even though the out-of-domain system initially yields lower BLEU scores.

The medical translation task represents a different situation, in which a large-scale system is built from various but domain-related corpora, among which one is used to train the CSTM. Nevertheless, results reported in Table 5.8 exhibit a similar trend. For both conditions, the discriminative training gives a significant improvement, up to 0.7 BLEU points over the model only trained with NCE, and up to 1.7 BLEU points over the baseline system. Arguably, the difference between two conditions is much smaller than what has been observed with the TED Talks task, due to the fact that Patent-Abstract corpus (which is used to discriminatively train the CSTM) only corresponds to a small subset of the parallel data. However, the best strategy seems, here again, to exclude the data used for the CSTM from the data used to train the baseline SMT system. This strategy is most suitable in the Domain Adaptation framework, but still feasible in the training of large-scale systems where the amount of data exclusively reserved for the training of CSTMs is insignificant compared to the total amount of bilingual data, as reflected in Table 5.1.

5.5 Related work

Most recent work in domain adaptation for SMT focuses on the modification of the sufficient statistics by conventional discrete models (Foster and Kuhn, 2007; Bertoldi and Federico, 2009; Chen et al., 2013), or on data selection (Axelrod et al., 2011; Sennrich, 2012). Our work owes much to recent contributions in discriminative training and tuning methods for SMT systems that we have reviewed in Section 1.3. While perceptron-based learning has been first introduced in (Shen and Joshi, 2005; Liang et al., 2006), margin-based algorithms such as MIRA (Watanabe et al., 2007; Chiang et al., 2008; Cherry and Foster, 2012) are nowadays considered as more efficient to train feature rich translation systems. This property is particularly relevant in our case, since we intend to learn a large set of parameters ($\boldsymbol{\theta}$). Another trend considers the optimization problem as ranking (Shen et al., 2004; Shen and Joshi, 2005; Hopkins and May, 2011; Simianer et al., 2012). Note that the ranking task corresponds to the integration of CSTMs based on N-best list rescoring (Section 2.5.2). In this work, the proposed objective functions borrow from these two lines of research to both adapt the CSTM ($\boldsymbol{\theta}$) and tune its contribution ($\boldsymbol{\lambda}$) to the whole SMT system. This procedure can be considered as an instance of *discriminative integrated training* described in Section 1.3.2.

To the best of our knowledge, the most similar work on discriminative training or

adaptation of neural network models is (Gao et al., 2014). In this article, the authors propose to estimate a neural network-based phrase translation model towards the expected BLEU (the Minimum Risk Training in Section 1.3.1), while tuning $\boldsymbol{\lambda}$ by standard tools. Algorithm 5 is very similar to their optimization algorithm, except that in our case, the feature weights $\boldsymbol{\lambda}$ are regularly updated for a better and tighter integration of the CSTM into the SMT system. Moreover, their model only considers phrase pairs in isolation, while we use a probabilistic model of the joint distribution of sentence pairs. Expected BLEU training has also been applied to recurrent NNLM (Auli and Gao, 2014).

For ranking language models (Section 2.3.3), (Collobert and Weston, 2008; Collobert et al., 2011) also introduce a ranking-type objective function, but which aims only to estimate word embeddings in a multi-task learning framework. Furthermore, Socher et al. (2013) investigate the use of a max-margin criterion to train a recursive neural network for syntactic parsing. Interestingly, their model is also used to rerank N-best derivations generated by a conventional probabilistic context-free grammar. However, as showed in our experiments, the max-margin criterion alone is less adapted to SMT for lack of a truly reliable and unambiguous quality measure.

5.6 Conclusions

In this chapter, we have described a discriminative method for training and adaptation tasks of continuous-space translation models. Compared to the usual CLL training of CSTMs, this framework has several advantages that lie in a better correlation between training criteria and the translation performance, and in the training process that is capable of taking into account all component models of the system. In an adaptation context, this method improves over the traditional adaptation strategy which requires all models (including the CSTM) to be retrained, and in which the training of each model is performed separately from the others. Instead, our approach proposes to only adapt the CSTM with new in-domain data while keeping intact other system components.

The contributions of this chapter are two-fold. First, we define a set of new training criteria for CSTMs, which are derived from the discriminative tuning and training of SMT systems, described in Section 1.3. Two typical loss functions are the max-margin and pairwise ranking criteria. While the first one is based on the structured perceptron the goal of which is to discriminatively learn to give the highest model score to the oracle translation, the second one considers multiple pairs of hypotheses, and is combined with a max-margin criterion which emphasizes relevant candidate pairs. Second, we evaluate the discriminative framework and the impact of different aspects within several training and adaptation situations. Another peculiarity of this chapter is the use of discriminative criteria on top of the standard CSM structure trained by NCE algorithm.

The experimental results allow us to draw several outstanding conclusions. First, empirical evidence from the text translation track of IWSLT'2011 English-to-French task prove the efficiency of the proposed framework in the adaptation situation, and of the discriminative criteria compared to the CLL optimisation. This experiment confirms an observation in (Le et al., 2012a) that continuous-space models are promising for Domain Adaptation, especially through the proposed discriminative adaptation framework.

Second, initialization, in particular of the CSTM parameters, is an important aspect of the proposed discriminative procedure. By a careful initialization schema using the CLL criterion, we can achieve a mixture of information from various sources inside the final model : combination between n-gram-level and sentence-level indications, as well as of unsupervised and supervised learning. According to experimental evidence, the best result is obtained by concatenating two modes of learning : first, the model is pre-trained with CLL optimization (or its NCE approximation), then the discriminative training is performed as a fine-tuning phase.

Moreover, based on this best configuration, we have compared different discriminative criteria that have been described in Chapter 1. Two important features of these criteria have been identified as the main factors in making the proposed procedure successful : the use of margins via a cost function, and the critical set which considers multiple hypothesis pairs at once. In particular, it seems that the inconsistency between the update step of $\boldsymbol{\theta}$ and the tuning of $\boldsymbol{\lambda}$ is a factor that can explain the bad performance of some approaches.

Finally, the impact of the baseline SMT system has also been investigated. Comparing between training and adaptation tasks, adaptation with discriminative method has a greater impact on the final translation performance. The best strategy consists of excluding the data for CSTM training from the parallel data used for the baseline SMT system. This is particularly suited to adaptation situations, but also to large-scale training tasks where training the CSTM involves only a small fraction of the total training data.

Conclusion

The work described in this dissertation investigates different methods and strategies to learn continuous-space models (CSMs) for Statistical Machine Translation (SMT). In most cases, the use of CSMs in Machine Translation (MT) is based on the encoding of existing knowledge under the form of *feature functions*. This *log-linear* approach generally allows the system to easily incorporate models of various natures and complexities, which forms the basis for *incremental* improvements and developments of SMT systems reported in the literature. As a consequence, this trend has led to a situation where the set of feature functions has become so large and diversified that the task of finding an optimal contribution of each component function has become extremely complicated. The development and consolidation of algorithms for this *tuning* step of the log-linear model is hence primordial for current SMT architectures, and remains until now an active research direction, especially for large-scale settings (Cherry and Foster, 2012; Yu et al., 2013; Zhao et al., 2014). An important feature characterizing this research area is the design of training criteria that tightly incorporate MT quality metrics, and of training procedures that involve parameters not only from the log-linear model, but from the whole SMT system.

The development of CSMs for MT also follows this incremental improvement of SMT systems. [15] This line of research has initiated with the study of monolingual NNLMs (Bengio et al., 2001), then later also includes bilingual CSTMs (Schwenk et al., 2007; Le et al., 2012a). These models have boosted the performance of SMT systems, as measured in terms of MT metrics (such as BLEU), but are also considered as being extremely expensive in training and testing, especially for large-vocabulary systems. Moreover, another problem which originates in the training of traditional language models is the inconsistency between the objective function defined for parameter estimation and the evaluation metrics derived from the application. NNLMs and CSTMs are often trained to optimize the CLL corresponding to an intrinsic evaluation using *perplexity*; however this measure has been shown to only loosely correlate with the performance of the CSM within the final application. It is likely that the CSM parameters are not optimally trained to yield the best performance for the translation system.

[15] In spite of the fact that CSMs now can be used as an independent approach for SMT (Sutskever et al., 2014; Bahdanau et al., 2014), integrating a CSM within a SMT system remains a popular and much simpler method to obtain good translation performance, and to evaluate the efficiency of the CSM.

Summary of my contributions

The contributions of this dissertation consist of proposing and evaluating solutions for these issues in order to obtain a better translation performance of the overall system. The cost of training and testing a large-vocabulary CSM can be reduced by using the SOUL output layer (Section 3.1), or the standard model structure trained using the NCE algorithm (Section 3.2). In Chapter 3, these two methods have proven to be equivalently efficient approaches to speed up the building of CSMs for SMT. While the training of SOUL models directly optimizes the CLL, the NCE algorithm belongs to the category of sampling-based procedures which approximately estimate the gradient of this objective function based on a small subset of negative examples from the output (or target) vocabulary. Within this category, the NCE CSMs (Mnih and Teh, 2012; Vaswani et al., 2013), even though also derived from *Importance Sampling* algorithm, are more advantageous than *Biased Importance Sampling* approach (Bengio et al., 2003b; Bengio and Senécal, 2008). Its corresponding training is more stable and delivers *self-normalized* scores which speed up the testing phase. Chapter 3 also puts forward an appropriate noise distribution which guarantees the convergence of the NCE training. In general, a unigram distribution can be used, subject to the condition that this noise distribution fits the one estimated on the data used to train the CSM.

The SOUL structure and NCE algorithm help to reduce the complexity of each training iteration, but the number of such iterations depends on how fast the convergence is achieved. To improve the convergence speed during the training phase, appropriately adapting the learning rates is shown to be an important issue (Chapter 4), in particular for the NCE training which is only an approximation of the CLL optimization. Two kinds of adaptive learning rates have been investigated : *global learning rate* scenarios, and *local learning rate* regimes which adjust one hyper-parameter for each block of model parameters (*Block-AdaGrad*). We have studied both the convergence speed and the sensibility of each method to other hyper-parameters (such as the first learning rate value and decay coefficient). Our experimental results have shown that the proposed *Down-Block-AdaGrad* achieves the best performance (in terms of perplexity) in various training scenarios involving SOUL and NCE models.

Instead of only training the CSMs towards an *intrinsic* perplexity measure, our final goal is to improve their effectiveness in the SMT application. To this end, a discriminative procedure has been proposed (Algorithm 5) for training and adaptation scenarios, which draws inspiration from the discriminative tuning and training of SMT systems (Section 1.3). For training tasks, the method has two advantages compared to the usual CLL optimization : a better correlation between the training criterion and final performance, and the consideration of the other components of the system during the training process. These experiments also prove the efficiency of discriminative Domain Adaptation, which requires only to adapt the CSTM while keeping other components intact. In terms of training criteria, objective functions which have been used for the Discriminative tuning of SMT systems (such as hinge loss, pairwise ranking or ListNet algorithm) can also be used in the proposed procedure. However, experimental evidence (Section 5.4.3) has allowed us to identify two important features for improving the performance : the use of margins via an appropriate cost function, and the inclusion of multiple hypothesis pairs from the search space. Other aspects have also been investigated, such as initialization

issues, or the relationship between the baseline system and the training data used for CSMs, which decides whether we are in a training or adaptation situation (Section 5.4.4). As a general conclusion, this discriminative framework is shown to outperform the CSMs which have been trained only by the CLL optimization (or the NCE algorithm), both in adaptation as well as in training situations. This result suggests rooms for improvements by adopting training criteria which correlates better with the actual error metric, whereas work solely based on perplexity may result in disappointing performance. The experimental study also presents more significant gains of this method in adaptation than in training situations, due to the fact that in the latter, the SMT system, which has been trained on training data, often produces hypotheses with over-optimistic BLEU scores compared to the ones that are generated for the test set. It is of general rule to exclude the training data dedicated for CSMs from the training data of the SMT system. Its use is hence most suited to adaptation tasks, but also to several training tasks where the total parallel training data is sufficiently large such that the impact of this exclusion from the SMT system training data is negligible.

Future work

To have a complete understanding of different discriminative training criteria described in this dissertation, a direct extension of the comparative study in Section 5.4.3 is to use various tuning algorithms other than the batch N-best MIRA. The role in the final performance of **the consistency between the training and tuning criteria** needs to be confirmed by future experiments.

The discriminative training framework defines **a schema for iteratively adding multiple component models into the SMT system**. Each model is learnt using the baseline N-best lists rescored with previously added models, in the hope that it will capture complementary information and correct errors of the previous pass. This scenario would give a comprehensive training procedure where the impact of CSMs remains measurable, instead of the currently widely-used method where the performance of CLL-trained CSMs integrated within the systems is unknown before the testing phase. Weighting differently training examples, similarly to the *boosting* algorithm (Freund and Schapire, 1995) also seems to be a promising approach. Moreover, our experiments so far have been based on the feed-forward neural network. A replication of these experiments with other model structures, such as the recurrent and convolutional neural networks can also be foreseen.

An interesting research direction is to investigate **the role of word embeddings** in the discriminative framework proposed in Chapter 5. Previous works (Collobert and Weston, 2008; Le, 2012; Huang et al., 2012a) have provided multiple evidences that the *relatedness* between words are well reflected through their embeddings; this property is considered as a source of improvements of CSMs compared to discrete models. The performance of our discriminative training has been boosted by pre-training word continuous representations (Section 5.4.2); it suggests that initializing the models using unlabelled data (such as the training of NNLMs) is always useful. The exploitation of such feature vectors on other tasks, such as word or phrase *alignment model* will be a natural extension.

In the scope of this dissertation, experiments have been carried out based on the N-best rescoring framework (Section 2.5.2). This approach can be extended in various ways. On the one hand, it would be interesting to **evaluate alternative ways to integrate discriminatively trained CSMs**, such as a more direct integration into the decoder. The use of such models within decoding is facilitated by the NCE model which provides a *self-normalized* output layer with fast inference (Vaswani et al., 2013). Several speeding-up techniques such as *pre-computing* the hidden layers (Devlin et al., 2014) or the use of *cache* (Vaswani et al., 2013) could be necessary to obtain a computationally efficient decoder.

On the other hand, **other training procedures could be derived from various integration modes**. It is important to notice that the *max-margin* objective function described in this work differs from the usual *perceptron* algorithm by the fact that references are often unreachable in SMT; oracle translations are used instead at the cost however of a loss of the theoretical guarantees. This issue has been discussed in (Huang et al., 2012b) with the *violation-fixing perceptron* framework, and applications have been described in (Yu et al., 2013), or in (Zhao et al., 2014) for hierarchical phrase-based systems. This idea can be used to discriminatively train CSTMs based on reference translations instead of the oracle translations extracted from N-best lists as described in this dissertation.

***Neural Machine Translation* (NMT) framework** is a recently proposed approach for SMT in which a neural network-based model is trained to directly generate translations from the source sentences without any external linguistic knowledge and hand-engineered features. While the work described in this dissertation considers CSMs as a mean to boost the performance of the actual SMT systems, NMT proposes CSMs as an independent approach for SMT (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014). In spite of this divergence, a combination between NMT and the ideas proposed in this work can be formed in different ways. First, it is interesting to assess the performance of our discriminatively trained CSMs within the NMT framework. Indeed, the SOUL structure provides an efficient solution for generating texts from large vocabularies (Allauzen et al., 2013) : a path from the root to a leaf of the clustering tree can be sampled by a sequential procedure. Such generation capacity is the core of various NMT approaches, but they often face issues related to large vocabularies (Jean et al., 2015). It seems that CSTMs with SOUL structure can be modified to work around their dependency on phrasal alignments, hence will be capable of generating translations in a very efficient way. Second, the discriminative training framework described in this dissertation is particularly important to the current NMT systems for which the training mostly relies on the maximization of CLL. In such situation, the *violation-fixing perceptron* framework (Huang et al., 2012b) can be used to lift the dependency on oracle translations from a baseline system, hence making this discriminative approach a complete and feasible solution for NMT.

List of publications

Do, Q.-K., Allauzen, A., and Yvon, F. (2015a). Apprentissage discriminant des modèles continus de traduction. In *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles*, pages 267–278, Caen, France. Association pour le Traitement Automatique des Langues.

Do, Q.-K., Allauzen, A., and Yvon, F. (2015b). A discriminative training procedure for continuous translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1046–1052, Lisbon, Portugal. Association for Computational Linguistics.

Niehues, J., DO, Q.-K., Allauzen, A., and Waibel, A. (2015). Listnet-based MT Rescoring. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 248–255, Lisbon, Portugal. Association for Computational Linguistics.

Marie, B., Allauzen, A., Burlot, F., Do, Q.-K., Ive, J., Knyazeva, E., Labeau, M., Lavergne, T., Löser, K., Pécheux, N., and Yvon, F. (2015). LIMSI@WMT'15 : Translation Task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 145–151, Lisbon, Portugal. Association for Computational Linguistics.

Ha, T.-L., DO, Q.-K., Cho, E., Niehues, J., Allauzen, A., Yvon, F., and Waibel, A. (2015). The KIT-LIMSI translation system for WMT 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 120–125, Lisbon, Portugal. Association for Computational Linguistics.

Do, Q.-K., Allauzen, A., and Yvon, F. (2014a). Discriminative adaptation of continuous space translation models. In *International Workshop on Spoken Language Translation (IWSLT 2014)*, Lake Tahoe, USA.

Do, Q.-K., Allauzen, A., and Yvon, F. (2014b). Modèle de langue neuronal: comparaison de différentes stratégies d'apprentissage. In *Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2014)*, pages 256–267, Marseille, France.

Do, Q.-K., Herrmann, T., Niehues, J., Allauzen, A., Yvon, F., and Waibel, A. (2014c). The KIT-LIMSI Translation System for WMT 2014. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 84–89, Baltimore, Maryland, USA.

Pécheux, N., Gong, L., Do, Q.-K., Marie, B., Ivanishcheva, Y., Allauzen, A., Lavergne, T., Niehues, J., Max, A., and Yvon, F. (2014). Limsi@ wmt'14 medical translation task. *ACL 2014*, page 246.

Allauzen, A., Pécheux, N., Do, Q.-K., Dinarelli, M., Lavergne, T., Max, A., Le, H.-s., and Yvon, F. (2013). LIMSI @ WMT13. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 62–69, Sofia, Bulgaria.

Peitz, S., Mansour, S., Huck, M., Freitag, M., Ney, H., Cho, E., Herrmann, T., Mediani, M., Niehues, J., Waibel, A., Allauzen, A., Khanh Do, Q., Buschbeck, B., and Wandmacher, T. (2013). Joint WMT 2013 submission of the QUAERO project. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 185–192, Sofia, Bulgaria. Association for Computational Linguistics.

Bibliography

Aho, A. V. and Ullman, J. D. (1969). Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37–56.

Allauzen, A., Pécheux, N., Do, Q.-K., Dinarelli, M., Lavergne, T., Max, A., Le, H.-s., and Yvon, F. (2013). LIMSI @ WMT13. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 62–69, Sofia, Bulgaria.

Alshawi, H., Bangalore, S., and Douglas, S. (2000). Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26(1):45–60.

Andrés-Ferrer, J. and Juan, A. (2009). A phrase-based hidden semi-markov approach to machine translation. In *13th Annual Conference of the European Association for Machine Translation*, page 168.

Auli, M., Galley, M., Quirk, C., and Zweig, G. (2013). Joint language and translation modeling with recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1044–1054.

Auli, M. and Gao, J. (2014). Decoder integration and expected BLEU training for recurrent neural network language models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*, pages 136–142.

Axelrod, A., He, X., and Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 355–362.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72.

Bansal, M., Quirk, C., and Moore, R. C. (2011). Gappy phrasal alignment by agreement. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1308–1317. Association for Computational Linguistics.

Bell, T. C., Cleary, J. G., and Witten, I. H. (1990). *Text compression*. Prentice-Hall, Inc.

Bellegarda, J. R. (1997). A latent semantic analysis framework for large-span language modeling. In *EUROSPEECH*.

Bellegarda, J. R. (2000). Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, 88(8):1279–1296.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer.

Bengio, Y., De Mori, R., Flammia, G., and Kompe, R. (1992). Global optimization of a neural network-hidden markov model hybrid. *Neural Networks, IEEE Transactions on*, 3(2):252–259.

Bengio, Y., Ducharme, R., and Vincent, P. (2001). A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, pages 932–938.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003a). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Bengio, Y. and Senécal, J.-S. (2008). Adaptive importance sampling to accelerate training of a neural probabilistic language model. *Neural Networks, IEEE Transactions on*, 19(4):713–722.

Bengio, Y., Senécal, J.-S., et al. (2003b). Quick training of probabilistic neural nets by importance sampling. In *AISTATS Conference*.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Bertoldi, N. and Federico, M. (2009). Domain adaptation for statistical machine translation with monolingual resources. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 182–189, Athens, Greece.

Bilmes, J. A. and Kirchhoff, K. (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*, pages 4–6. Association for Computational Linguistics.

Birch, A., Huck, M., Durrani, N., Bogoychev, N., and Koehn, P. (2014). Edinburgh SLT and MT System Description for the IWSLT 2014 Evaluation. In Federico, M., Stücker, S., and Yvon, F., editors, *Proceedings of the eleventh International Workshop on Spoken Language Translation (IWSLT)*, pages 49–56.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Blitzer, J. (2008). *Domain Adaptation of Natural Language Processing Systems*. PhD thesis, University of Pennsylvania.

Blunsom, P., Cohn, T., and Osborne, M. (2008). A discriminative latent variable model for statistical machine translation. In *ACL*, pages 200–208.

Blunsom, P., Kalchbrenner, N., et al. (2013). Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the 2013 Workshop on Continuous Vector Space Models and their Compositionality*. Proceedings of the 2013 Workshop on Continuous Vector Space Models and their Compositionality.

Blunsom, P. and Osborne, M. (2008). Probabilistic inference for machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 215–223. Association for Computational Linguistics.

Boden, M. (2001). A guide to recurrent neural networks and backpropagation.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

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

Callison-Burch, C., Fordyce, C., Koehn, P., Monz, C., and Schroeder, J. (2007). (Meta-) Evaluation of Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158. Association for Computational Linguistics.

Callison-Burch, C., Osborne, M., and Koehn, P. (2006). Re-evaluating the role of BLEU in Machine Translation research. In *EACL*, volume 6, pages 249–256.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM.

Carroll, G. and Charniak, E. (1992). *Two experiments on learning probabilistic dependency grammars from corpora*. Department of Computer Science, Univ.

Casacuberta, F. and Vidal, E. (2004). Machine translation with inferred stochastic finite-state transducers. *Computational Linguistics*, 30(3):205–225.

Cer, D., Jurafsky, D., and Manning, C. D. (2008). Regularization and search for minimum error rate training. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 26–34. Association for Computational Linguistics.

Chelba, C., Engle, D., Jelinek, F., Jimenez, V., Khudanpur, S., Mangu, L., Printz, H., Ristad, E., Rosenfeld, R., Stolcke, A., et al. (1997). Structure and performance of a dependency language model. In *EUROSPEECH*. Citeseer.

Chelba, C. and Jelinek, F. (2000). Recognition performance of a structured language model. *arXiv preprint cs/0001022*.

Chen, B., Kuhn, R., and Foster, G. (2013). Vector space model for adaptation in statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1285–1293.

Chen, S. F., Beeferman, D., and Rosenfeld, R. (1998). Evaluation metrics for language models.

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics.

Chen, S. F. and Rosenfeld, R. (2000). A survey of smoothing techniques for me models. *Speech and Audio Processing, IEEE Transactions on*, 8(1):37–50.

Cherry, C. and Foster, G. (2012). Batch tuning strategies for statistical machine translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 427–436.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational linguistics*, 33(2):201–228.

Chiang, D. (2012). Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13(1):1159–1187.

Chiang, D., Knight, K., and Wang, W. (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*, pages 218–226. Association for Computational Linguistics.

Chiang, D., Marton, Y., and Resnik, P. (2008). Online large-margin training of syntactic and structural translation features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 224–233. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Collins, M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–8.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

Crego, J. M. and Mariño, J. B. (2006). Improving statistical MT by coupling reordering and decoding. *Machine Translation*, 20(3):199–215.

Crego, J. M. and Yvon, F. (2010). Factored bilingual n-gram language models for statistical machine translation. *Machine Translation*, 24(2):159–175.

Crego, J. M., Yvon, F., and Mariño, J. B. (2011). N-code: an open-source bilingual N-gram SMT toolkit. *Prague Bulletin of Mathematical Linguistics*, 96:49–58.

Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42.

Daumé III, H. and Marcu, D. (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

DeNero, J., Bouchard-Côté, A., and Klein, D. (2008). Sampling alignment structure under a bayesian translation model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 314–323. Association for Computational Linguistics.

DeNero, J., Gillick, D., Zhang, J., and Klein, D. (2006). Why generative phrase models underperform surface heuristics. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 31–38. Association for Computational Linguistics.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland.

Do, Q.-K., Allauzen, A., and Yvon, F. (2014a). Discriminative adaptation of continuous space translation models. In *International Workshop on Spoken Language Translation (IWSLT 2014)*, Lake Tahoe, USA.

Do, Q.-K., Allauzen, A., and Yvon, F. (2014b). Modèle de langue neuronal: comparaison de différentes stratégies d'apprentissage. In *Conférence sur le Traitement Automatique des Langues Naturelles (TALN 2014)*, pages 256–267, Marseille, France.

Do, Q.-K., Allauzen, A., and Yvon, F. (2015a). Apprentissage discriminant des modèles continus de traduction. In *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles*, pages 267–278, Caen, France. Association pour le Traitement Automatique des Langues.

Do, Q.-K., Allauzen, A., and Yvon, F. (2015b). A discriminative training procedure for continuous translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1046–1052, Lisbon, Portugal. Association for Computational Linguistics.

Do, Q.-K., Herrmann, T., Niehues, J., Allauzen, A., Yvon, F., and Waibel, A. (2014c). The KIT-LIMSI Translation System for WMT 2014. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 84–89, Baltimore, Maryland, USA.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Dyer, C. and Resnik, P. (2010). Context-free reordering, finite-state translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 858–866. Association for Computational Linguistics.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Federico, M., Stüker, S., Bentivogli, L., Paul, M., Cettolo, M., Herrmann, T., Niehues, J., and Moretti, G. (2012). The IWSLT 2011 evaluation campaign on automatic talk translation. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA).

Feng, Y. and Cohn, T. (2013). A Markov model of machine translation using non-parametric Bayesian inference. In *ACL (1)*, pages 333–342.

Fishel, M., Sennrich, R., Popović, M., and Bojar, O. (2012). Terrorcat: a translation error categorization-based MT quality metric. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 64–70. Association for Computational Linguistics.

Foster, G. and Kuhn, R. (2007). Mixture-model adaptation for SMT. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 128–135, Prague, Czech Republic.

Foster, G. and Kuhn, R. (2009). Stabilizing minimum error rate training. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 242–249.

Foster, G., Kuhn, R., and Johnson, J. H. (2006). Phrase table smoothing for statistical machine translation.

Freund, Y. and Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer.

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

Gao, J. and He, X. (2013). Training MRF-based phrase translation models using gradient ascent. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 450–459, Atlanta, Georgia. Association for Computational Linguistics.

Gao, J., He, X., Yih, W.-t., and Deng, L. (2014). Learning continuous phrase representations for translation modeling. In *Proc. ACL*.

Gers, F. A., Schraudolph, N. N., and Schmidhuber, J. (2003). Learning precise timing with LSTM recurrent networks. *The Journal of Machine Learning Research*, 3:115–143.

Gildea, D. and Hofmann, T. (1999). Topic-based language models using EM. In *Proceedings of the 6th European Conference on Speech Communication and Technology (EUROSPEECH-99)*, pages 2167–2170, Budapest.

Gimpel, K. and Smith, N. A. (2010). Softmax-margin CRFs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736. Association for Computational Linguistics.

Gimpel, K. and Smith, N. A. (2012). Structured ramp loss minimization for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231. Association for Computational Linguistics.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264.

Goodman, J. T. (2001). A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.

Green, S., Wang, S., Cer, D., and Manning, C. D. (2013). Fast and adaptive online training of feature-rich translation models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–321.

Gubbins, J. and Vlachos, A. (2013). Dependency language models for sentence completion. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1405–1410, Seattle, Washington, USA. Association for Computational Linguistics.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Teh, Y. W. and Titterington, M., editors, *Proceedings of th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 297–304.

Ha, T.-L., DO, Q.-K., Cho, E., Niehues, J., Allauzen, A., Yvon, F., and Waibel, A. (2015). The KIT-LIMSI translation system for WMT 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 120–125, Lisbon, Portugal. Association for Computational Linguistics.

He, X. and Deng, L. (2012). Maximum expected BLEU training of phrase and lexicon translation models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 292–301. Association for Computational Linguistics.

Hermann, K. M. and Blunsom, P. (2013). Multilingual distributed representations without word alignment. *arXiv preprint arXiv:1312.6173*.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hopkins, M. and May, J. (2011). Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK.

Hu, Y., Auli, M., Gao, Q., and Gao, J. (2014). Minimum translation modeling with recurrent neural networks. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 20–29.

Huang, E., Socher, R., Manning, C., and Ng, A. (2012a). 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)*, pages 873–882, Jeju Island, Korea.

Huang, L. (2008). Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

Huang, L., Fayong, S., and Guo, Y. (2012b). Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151. Association for Computational Linguistics.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015). On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Jelinek, F. (1976). Speech recognition by statistical methods. *Proceedings of the IEEE*, 64:532–556.

Jelinek, F. (1980). Interpolated estimation of Markov source parameters from sparse data. *Pattern recognition in practice*.

Joty, S., Guzmán, F., Màrquez, L., and Nakov, P. (2014). Discotk: Using discourse structure for machine translation evaluation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 402–408.

Juang, B.-H., Chou, W., and Lee, C.-H. (1996). Statistical and discriminative methods for speech recognition. In *Automatic Speech and Speaker Recognition*, pages 109–132. Springer.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709, Seattle, Washington, USA.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):400–401.

Kneser, R. and Ney, H. (1993). Improved clustering techniques for class-based statistical language modelling. In *Eurospeech*, volume 93, pages 973–76.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Koehn, P. and Hoang, H. (2007). Factored translation models. In *EMNLP-CoNLL*, pages 868–876.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (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*, pages 177–180. Association for Computational Linguistics.

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

Kong, A., Liu, J. S., and Wong, W. H. (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American statistical association*, 89(425):278–288.

Koo, T. and Collins, M. (2005). Hidden-variable models for discriminative reranking. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 507–514. Association for Computational Linguistics.

Lang, K. J., Waibel, A. H., and Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural networks*, 3(1):23–43.

Lau, R., Rosenfeld, R., and Roukos, S. (1993). Trigger-based language models: A maximum entropy approach. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 45–48. IEEE.

Lavergne, T., Allauzen, A., and Yvon, F. (2013). Un cadre d'apprentissage intégralement discriminant pour la traduction statistique. *TALN-RÉCITAL 2013*, page 450.

Lavergne, T., Crego, J. M., Allauzen, A., and Yvon, F. (2011). From n-gram-based to CRF-based translation models. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 542–553. Association for Computational Linguistics.

Le, H. S. (2012). *Continuous space models with neural networks in natural language processing*. PhD thesis, Université Paris Sud-Paris XI.

Le, H.-S., Allauzen, A., and Yvon, F. (2012a). Continuous space translation models with neural networks. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 39–48, Montréal, Canada.

Le, H.-S., Allauzen, A., and Yvon, F. (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*, pages 1–10. Association for Computational Linguistics.

Le, H.-S., Oparin, I., Allauzen, A., Gauvain, J.-L., and Yvon, F. (2011). Structured output layer neural network language model. In *Proceedings of the International Conference on Audio, Speech and Signal Processing*, pages 5524–5527.

Le, H.-S., Oparin, I., Allauzen, A., Gauvain, J.-L., and Yvon, F. (2013). Structured output layer neural network language models for speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(1):197–206.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Liang, P., Bouchard-Côté, A., Klein, D., and Taskar, B. (2006). An end-to-end discriminative approach to machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 761–768.

Lin, C.-Y. and Och, F. J. (2004). Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, page 501. Association for Computational Linguistics.

Lin, T., Horne, B. G., Tiňo, P., and Giles, C. L. (1996). Learning long-term dependencies in narx recurrent neural networks. *Neural Networks, IEEE Transactions on*, 7(6):1329–1338.

Lo, C.-k. and Wu, D. (2013). MEANT at WMT 2013: A tunable, accurate yet inexpensive semantic frame based MT evaluation metric. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 422–428.

Lopez, A. (2008). Statistical machine translation. *ACM Computing Surveys (CSUR)*, 40(3):8.

Macherey, W., Och, F. J., Thayer, I., and Uszkoreit, J. (2008). Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 725–734. Association for Computational Linguistics.

Marcu, D. and Wong, W. (2002). A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 133–139. Association for Computational Linguistics.

Martens, J. (2010). Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742.

Martens, J. and Sutskever, I. (2011). Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1033–1040.

Martins, A. F., Gimpel, K., Smith, N. A., Xing, E. P., Figueiredo, M. A., and Aguiar, P. M. (2010). Learning structured classifiers with dual coordinate ascent. Technical report, DTIC Document.

McCallum, A., Bellare, K., and Pereira, F. (2012). A conditional random field for discriminatively-trained finite-state string edit distance. *arXiv preprint arXiv:1207.1406*.

McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–98.

Miikkulainen, R. and Dyer, M. G. (1991). Natural language processing with modular PDP networks and distributed lexicon. *Cognitive Science*, 15(3):343–399.

Mikolov, T., Deoras, A., Povey, D., Burget, L., and Černockỳ, J. (2011a). Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 196–201. IEEE.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Mikolov, T., Kombrink, S., Burget, L., Cernocký, J., and Khudanpur, S. (2011b). Extensions of recurrent neural network language model. In *Proceedings of ICASSP*, pages 5528–5531.

Mikolov, T. and Zweig, G. (2012). Context dependent recurrent neural network language model. In *SLT*, pages 234–239.

Mnih, A. and Hinton, G. E. (2007). Three new graphical models for statistical language modelling. In *Proceedings of the International Conference of Machine Learning (ICML)*, pages 641–648.

Mnih, A. and Hinton, G. E. (2008). A scalable hierarchical distributed language model. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, volume 21, pages 1081–1088.

Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the International Conference of Machine Learning (ICML)*.

Moore, R. C. and Quirk, C. (2008). Random restarts in minimum error rate training for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 585–592. Association for Computational Linguistics.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252. Citeseer.

Mozer, M. C. (1993). Induction of multiscale temporal structure. *Advances in neural information processing systems*, pages 275–275.

Nakamura, M., Maruyama, K., Kawabata, T., and Kiyohiro, S. (1990). Neural network approach to word category prediction for english texts. In *Proceedings of the 13th conference on Computational linguistics (COLING)*, volume 3, pages 213–218.

Nakov, P., Guzmán, F., and Vogel, S. (2012). Optimizing for Sentence-Level BLEU+ 1 yields short translations. In *COLING*, pages 1979–1994.

Ney, H., Essen, U., and Kneser, R. (1994). On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1–38.

Niehues, J., DO, Q.-K., Allauzen, A., and Waibel, A. (2015). Listnet-based MT Rescoring. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 248–255, Lisbon, Portugal. Association for Computational Linguistics.

Niehues, J. and Waibel, A. (2012). Continuous space language models using restricted Boltzmann machines. In *Proceedings of International Workshop on Spoken Language Translation (IWSLT)*, pages 164–170, Hong-Kong, China.

Och, F. J. (1999). An efficient method for determining bilingual word classes. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 71–76. Association for Computational Linguistics.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302. Association for Computational Linguistics.

Och, F. J., Tillmann, C., Ney, H., et al. (1999). Improved alignment models for statistical machine translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

Ollivier, Y. (2013). Riemannian metrics for neural networks. *arXiv preprint arXiv:1303.0818*.

Ortiz, L. E. and Kaelbling, L. P. (2000). Adaptive importance sampling for estimation in structured domains. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 446–454. Morgan Kaufmann Publishers Inc.

Paccanaro, A. and Hinton, G. E. (2000). Extracting distributed representations of concepts and relations from positive and negative propositions. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 2, pages 259–264. IEEE.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modelling. *Computer Speech & Language*, 10(3):187–228.

Rosenfeld, R. (2000). Two decades of statistical language modeling: Where do we go from here?

Rosti, A.-V. I., Zhang, B., Matsoukas, S., and Schwartz, R. (2011). Expected BLEU training for graphs: BBN system description for WMT11 system combination task. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 159–165. Association for Computational Linguistics.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

Sarikaya, R., Deng, Y., Kingsbury, B. E. D., and Gao, Y. (2008). Machine translation in continuous space. US Patent App. 12/054,636.

Schaul, T., Zhang, S., and LeCun, Y. (2012). No more pesky learning rates. *arXiv preprint arXiv:1206.1106*.

Schlüter, R. and Ney, H. (2001). Model-based MCE bound to the true Bayes' error. *Signal Processing Letters, IEEE*, 8(5):131–133.

Schmidhuber, J. (1992). Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242.

Schmidhuber, J. and Heil, S. (1996). Sequential neural text compression. *Neural Networks, IEEE Transactions on*, 7(1):142–146.

Schwenk, H. (2007). Continuous space language models. *Computer Speech and Language*, 21(3):492–518.

Schwenk, H. and Gauvain, J.-L. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–765. IEEE.

Schwenk, H. and Gauvain, J.-L. (2004). Neural network language models for conversational speech recognition. In *INTERSPEECH*.

Schwenk, H., R. Costa-jussa, M., and R. Fonollosa, J. A. (2007). Smooth bilingual *n*-gram translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 430–438, Prague, Czech Republic.

Schwenk, H., Rousseau, A., and Attik, M. (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*, pages 11–19. Association for Computational Linguistics.

Seide, F., Li, G., and Yu, D. (2011). Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. In *Interspeech*, pages 437–440.

Senior, A., Heigold, G., Ranzato, M., and Yang, K. (2013). An empirical study of learning rates in deep neural networks for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6724–6728. IEEE.

Sennrich, R. (2012). Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the European chapter of the Association for Computational Linguistics (EACL)*, pages 539–549.

Shen, L. and Joshi, A. K. (2005). Ranking and reranking with perceptron. *Machine Learning*, 60(1-3):73–96.

Shen, L., Sarkar, A., and Och, F. J. (2004). Discriminative reranking for machine translation. In *HLT-NAACL*, pages 177–184.

Simianer, P., Riezler, S., and Dyer, C. (2012). Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 11–21.

Smith, D. A. and Eisner, J. (2006). Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 787–794. Association for Computational Linguistics.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231.

Socher, R., Bauer, J., Manning, C. D., and Andrew Y., N. (2013). Parsing with compositional vector grammars. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 455–465, Sofia, Bulgaria.

Sokolov, A., Wisniewski, G., and Yvon, F. (2012). Non-linear n-best list reranking with few features. In *Association for Machine Translation in the Americas*.

Stanojević, M. and Sima'an, K. (2014). BEER: BEtter Evaluation as Ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA. Association for Computational Linguistics.

Sundermeyer, M., Oparin, I., Gauvain, J.-L., Freiberg, B., Schluter, R., and Ney, H. (2013). Comparison of feedforward and recurrent neural network language models. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8430–8434. IEEE.

Sundermeyer, M., Schlüter, R., and Ney, H. (2012). LSTM Neural Networks for Language Modeling. In *INTERSPEECH*.

Sundermeyer, M., Tüske, Z., Schlüter, R., and Ney, H. (2014). Lattice decoding and rescoring with long-span neural network language models. In *Proc. of Interspeech*, pages 661–665.

Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, NIPS*27, pages 3104–3112, Montréal, Canada.

Tam, Y.-C. and Schultz, T. (2005). Dynamic language model adaptation using variational bayes inference. In *INTERSPEECH*, pages 5–8.

Tam, Y.-C. and Schultz, T. (2006). Unsupervised language model adaptation using latent semantic marginals. In *INTERSPEECH*.

Taskar, B., Klein, D., Collins, M., Koller, D., and Manning, C. D. (2004). Max-margin parsing. In *EMNLP*, volume 1, page 3. Citeseer.

Tillmann, C. (2004). A unigram orientation model for statistical machine translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 101–104.

Tillmann, C., Vogel, S., Ney, H., and Zubiaga, A. (1997). A dp based search using monotone alignments in statistical translation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 289–296. Association for Computational Linguistics.

Tillmann, C. and Zhang, T. (2006). A discriminative global training algorithm for statistical mt. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 721–728. Association for Computational Linguistics.

Tran, K. M., Bisazza, A., and Monz, C. (2014). Word translation prediction for morphologically rich languages with bilingual neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1676–1688, Doha, Qatar. Association for Computational Linguistics.

Turian, J., Ratinov, L.-A., and Bengio, Y. (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*, pages 384–394, Uppsala, Sweden.

Turian, J. P., Shea, L., and Melamed, I. D. (2006). Evaluation of machine translation and its evaluation. Technical report, DTIC Document.

Vaswani, A., Zhao, Y., Fossum, V., and Chiang, D. (2013). Decoding with large-scale neural language models improves translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1387–1392, Seattle, Washington, USA.

Vogel, S., Ney, H., and Tillmann, C. (1996). HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

Wang, S., Schuurmans, D., Peng, F., and Zhao, Y. (2003). Semantic n-gram language modeling with the latent maximum entropy principle. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–376. IEEE.

Ward, W. et al. (1990). The CMU air travel information service: Understanding spontaneous speech. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 127–129.

Watanabe, S., Iwata, T., Hori, T., Sako, A., and Ariki, Y. (2011). Topic tracking language model for speech recognition. *Computer Speech & Language*, 25(2):440–461.

Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. (2007). Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic.

White, J., O'Connell, T., and O'Mara, F. (1994). The ARPA MT evaluation methodologies: evolution, lessons, and future approaches. In *Proceedings of the 1994 Conference, Association for Machine Translation in the Americas*, pages 193–205.

Wisniewski, G. and Yvon, F. (2013). Fast large-margin learning for statistical machine translation. *International Journal of Computational Linguistics and Applications*, 4(2):45.

Witten, I. H. and Bell, T. C. (1991). The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *Information Theory, IEEE Transactions on*, 37(4):1085–1094.

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational linguistics*, 23(3):377–403.

Xiao, M. and Guo, Y. (2013). Domain adaptation for sequence labeling tasks with a probabilistic language adaptation model. In *Proceedings of The 30th International Conference on Machine Learning*, pages 293–301.

Xu, W. (2011). Towards optimal one pass large scale learning with averaged stochastic gradient descent. *arXiv preprint arXiv:1107.2490*.

Xu, W. and Rudnicky, A. I. (2000). Can artificial neural networks learn language models?

Yamada, K. and Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics.

Yang, N., Liu, S., Li, M., Zhou, M., and Yu, N. (2013). Word alignment modeling with context dependent deep neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 166–175, Sofia, Bulgaria.

Yu, H., Huang, L., Mi, H., and Zhao, K. (2013). Max-Violation Perceptron and Forced Decoding for Scalable MT Training. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Washington, USA*.

Zamora-Martinez, F., Castro-Bleda, M. J., and Schwenk, H. (2010). N-gram-based machine translation enhanced with neural networks for the french-english btec-iwslt'10 task. In *International Workshop on Spoken Language Translation (IWSLT) 2010*.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zens, R., Hasan, S., and Ney, H. (2007). A systematic comparison of training criteria for statistical machine translation. In *EMNLP-CoNLL*, pages 524–532.

Zens, R., Och, F. J., and Ney, H. (2002). Phrase-based statistical machine translation. In *KI '02: Proceedings of the 25th Annual German Conference on AI*, pages 18–32, London, UK. Springer-Verlag.

Zhang, H., Quirk, C., Moore, R. C., and Gildea, D. (2008). Bayesian learning of non-compositional phrases with synchronous parsing. In *ACL*, pages 97–105.

Zhang, J., Liu, S., Li, M., Zhou, M., and Zong, C. (2015). Beyond word-based language model in statistical machine translation. *arXiv preprint arXiv:1502.01446*.

Zhao, K., Huang, L., Mi, H., and Ittycheriah, A. (2014). Hierarchical MT training using max-violation perceptron. *Proceedings of ACL, Baltimore, Maryland, June*.