UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

Continuous Spaces in Statistical Machine Translation

Master Thesis

Máster en Inteligencia Artificial,
Reconocimiento de Formas e Imagen Digital

Author: Álvaro Peris Abril

Advisor: Francisco Casacuberta Nolla

September, 2015

Acknowledgements

The research leading to these results has received funding from the Ministerio de Economía y Sostenibilidad (MINECO) under grant RTC-2014-1466-4 and the Generalitat Valenciana under grant Prometeo/2014/030.

We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

Abstract

Classically, statistical machine translation relied on representations of words in a discrete space. Words and phrases were atomically represented as indices in a vector. In the last years, techniques for representing words and phrases in a continuous space have arisen. In this scenario, a word is represented in the continuous space as a real-valued, dense and low-dimensional vector. Statistical models can profit from this richer representation, since it is able to naturally take into account concepts such as semantic or syntactic relationships between words and phrases. This approach is encouraging, but it also entails new challenges.

In this work, a language model which relies on continuous representations of words is developed. Such model makes use of a bidirectional recurrent neural network, which is able to take into account both the past and the future context of words. Since the model is costly to train, the training dataset is reduced by using bilingual sentence selection techniques. Two selection methods are used and compared. The language model is then used to rerank translation hypotheses. Results show improvements on the translation quality.

Moreover, a new approach for machine translation has been recently proposed: The so-called neural machine translation. It consists in the sole use of a large neural network for carrying out the translation process. In this work, such novel model is compared to the existing phrase-based approaches of statistical machine translation.

Finally, the neural translation models are combined with diverse machine translation systems, in order to provide a consensus translation, which aim to improve the translation given by each single system.

Resumen

Los sistemas clásicos de traducción automática estadística están basados en representaciones de palabras en un espacio discreto. Palabras y segmentos se representan como índices en un vector. Durante los últimos años han surgido técnicas para realizar la representación de palabras y segmentos en un espacio continuo. En este escenario, una palabra se representa en el espacio continuo como un vector de valores reales, denso y de baja dimensión. Los modelos estadísticos pueden aprovecharse de esta representación más rica, puesto que incluye de forma natural conceptos semánticos o relaciones sintácticas entre palabras y segmentos. Esta aproximación es prometedora, pero también conlleva nuevos retos.

En este trabajo se desarrolla un modelo de lenguaje basado en representaciones continuas de palabras. Dicho modelo emplea una red neuronal recurrente bidireccional, la cual es capaz de considerar tanto el contexto pasado como el contexto futuro de las palabras. Debido a que este modelo es costoso de entrenar, se emplea un conjunto de entrenamiento reducido mediante técnicas de selección de frases bilingües. Se emplean y comparan dos métodos de selección. Una vez entrenado, el modelo se emplea para reordenar hipótesis de traducción. Los resultados muestran mejoras en la calidad de la traducción.

Por otro lado, recientemente se propuso una nueva aproximación a la traducción automática: la llamada traducción automática neuronal. Consiste en el uso exclusivo de una gran red neuronal para llevar a cabo el proceso de traducción. En este trabajo, este nuevo modelo se compara al paradigma actual de traducción basada en segmentos.

Finalmente, los modelos de traducción neuronales son combinados con otros sistemas de traducción automática, para ofrecer una traducción consensuada, que busca mejorar las traducciones individuales que cada sistema ofrece.

Resum

Els sistemes clàssics de traducció automàtica estadística es basen en representacions de paraules a un espai discret. Paraules i segments es representen com a índexs a un vector. Durant els últims anys, han sorgit tècniques per a realitzar la representació de paraules i segments a un espai continu. A aquest escenari, una paraula es representa a l'espai continu com a un vector de valors reals, dens i de baixa dimensió. Els models estadístics poden aprofitar-se d'aquesta representació més rica, donat que inclou de forma natural conceptes semàntics o relacions sintàctiques entre paraules i segments. Es tracta d'una aproximació prometedora, però comporta també nous reptes.

A aquest treball es desenvolupa un model de llenguatge basat en representacions contínues de paraules. El model utilitza una xarxa neuronal recurrent bidireccional, capaç de considerar tant el context passat com el context futur de les paraules. A causa de que el model és costós d'entrenar, s'empra un conjunt d'entrenament reduït mitjançant tècniques de selecció de frases bilingües. S'utilitzen i comparen dos mètodes de selecció. Un cop entrenat, el model s'empra per a reordenar hipòtesis de traducció. Els resultat mostren millores a la qualitat de la traducció.

D'altra banda, recentment es va proposar una nova aproximació a la traducció automàtica: l'anomenada traducció automàtica neuronal. Consisteix en l'ús exclusiu d'una gran xarxa neuronal per dur a terme el procés de traducció. Al present treball el nou model es compara amb el paradigma actual de traducció basada en segments.

Finalment, els models de traducció neuronals són combinats amb altres sistemes de traducció automàtica, per a oferir una traducció consensuada, que tracta millorar les traduccions individuals que cada sistema ofereix.

Contents

Preface

The human being is social by nature. Language is one of the most crucial tools developed by mankind for socializing and a fundamental form of communication. In our global society exists an important economic and social interest for translating languages, which help to overcome the walls that different languages eventually build between human relationships. Machine Translation (MT) is the computerized approach to the translation problem: It investigates the use of software with the aim of automatically translate text or speech from a source natural language to another target natural language.

This research area arose in 1949, from Warren Weaver's idea of apply some of the Information Theory ideas from Claude Shannon. During the 1950s and 1960s, the approaches for tackling the translation problem were polarized between the empirical (which use statistical methods) and the theoretical ones. Researchers were optimistic with regard to the machine translation and its capabilities (Koerner and Asher, 2014).

However, the results were unsatisfactory: Hardware limitations and the lack of effective programming methods made MT impracticable. The ALPAC (Automatic Language Processing Advisory Committee) report, released in 1966, concluded that machine translation was not worth and it presented an infeasible future. The area was then temporarily forsaken.

Ten years later, MT prospects grew once again as the rule-based machine translation approach was developed. It was based on the extraction of linguistic information from dictionaries and grammars by human translators, for generating the translations. Nevertheless, since 1989, the hegemony of theses systems was broken due to the irruption of the so-called corpus-based systems.

Such systems use sets of sentence-aligned translation examples, known as corpora or parallel texts, from one language to another. Such corpora are used to infer the translation of new source text. Among the corpus-based systems, the most successful approach is the statistical machine translation (SMT). In this scenario, the translation process is driven by statistical models. For SMT, large amount of parallel text with relevant information for the translation process is required. These corpora are used to estimate the parameters of the referred models. Once the parameters are estimated, the models infer the translation of new source text.

Classically, statistical models for MT treated the words as atomic units: They were indices in a vector. In the last years though, a new and encouraging approach for representing words has arisen. Such representation is based on continuous representation of words, i.e., words are represented by means of a real-valued, dense and low-dimensional vector. Statistical models are expected to profit from this richer representation of words, which can take into account

concepts like similarity and semantic relationships between words.

Neural networks provide a powerful tool for naturally handling continuous models. Moreover, recent advances in the deep learning field make this a promising approach, not only for machine translation, but also for many other tasks involving the natural language processing field.

This work is focused on the use and development of such continuous models, related to machine translation. This document is structured as follows: In Chapter 1, the statistical framework on which SMT relies is developed. In Chapter 2, neural languages models are introduced and an architecture for language modelling, based on a bidirectional recurrent neural network is presented. Chapter 3 presents the recently proposed neural approach for MT, based on the sole usage of neural networks for performing the translations. In addition, some methods for addressing the weaknesses of such technique are proposed. Since the training of those models is costly, two techniques for selecting the most relevant instances for training a model are depicted in Chapter 4. Performed experiments and results are shown and discussed in Chapter 5. Finally, conclusions about the work are drawn in Chapter 6.

Chapter 1

Statistical Machine Translation

Statistical Machine Translation tackles the translation problem using a statistic formalization thereof. The goal of SMT is, given a source sentence $x_1^J \equiv x_1 \dots x_j \dots x_J$, in the source language $\mathcal{F}$, to find its equivalent target sentence $y_1^I \equiv y_1 \dots y_i \dots y_I$, in the target language $\mathcal{E}$. From all possible sentence of $\mathcal{E}$, the one with the highest probability is chosen, according to the following expression:

$$\hat{y_1^I} = \arg\max_{I, y_1^I} \{Pr(y_1^I | x_1^J)\} \tag{1.1}$$

This equation requires to compute the joint probability of all possible sentence pair (x_1^J, y_1^I). Obviously, this computation is infeasible, since it would require to know all possible sentence in the language $\mathcal{E}$. For overcoming this issue, the first works on SMT tackled Equation 1.1 under a *generative* perspective. Generative models apply the Bayes decision rule to this expression: Considering that $Pr(x_1^J)$ is independent on y_1^I, Equation 1.1 can be rewritten as:

$$\hat{y_1^I} = \arg\max_{I, y_1^I} \{Pr(y_1^I) \cdot Pr(x_1^J | y_1^I)\} \tag{1.2}$$

Hence, $Pr(y_1^I | x_1^J)$ is decomposed in two different probability distributions. $Pr(y_1^I)$ determines how well the sentence in the target language is formed and is modelled by means of a *language model*. $Pr(x_1^J | y_1^I)$ determines, given the target sentence y_1^I, the chances of being translated into x_1^J. This distribution is modelled through a *translation model*.

Equation 1.2 is known as *fundamental equation of SMT* and it encapsulates the three main challenges of SMT (Brown et al., 1993):

1. Obtaining the language probability distribution: $Pr(y_1^I)$.
2. Obtaining the translation probability distribution: $Pr(x_1^J | y_1^I)$.
3. Finding and effective and efficient search search method for the string which maximizes the product, i.e., to efficiently compute $\arg\max_{I, y_1^I}$.

Since the real probability distributions are unknown, they must be estimated through (parametric) statistical models. Such models have a set of parameters Θ, linked to either a known probability density function or a probability mass function, expressed by $p(\cdot|\Theta)$. Given a set of observed samples, $\mathcal{X} = \{x_1, x_2, ..., x_N\}$, the *log-likelihood function* is defined as the logarithm of the probability of the observed data:

$$\mathcal{L}(\Theta, \mathcal{X}) = \log p(x_1, x_2, ..., x_N|\Theta) = \sum_{i=1}^{N} \log p(x_i|\Theta) \tag{1.3}$$

Typically, the estimation of Θ is carried out by means of the *maximum-likelihood* (ML) estimation method, which estimates the set of parameters Θ finding the value of Θ which maximizes $\mathcal{L}(\Theta, \mathcal{X})$. This value ($\hat{\Theta}$) is called the maximum-likelihood estimator of Θ:

$$\hat{\Theta} = \arg\max_{\Theta} \mathcal{L}(\Theta, \mathcal{X}) \tag{1.4}$$

Generally, both probability distributions which are involved in Equation 1.2, are modelled separately. Hence, exist two sets of parameters, those associated to the language model and those associated to the translation model:

$$\hat{\Theta}_{LM} = \arg\max_{\Theta_{LM}} \left\{ \sum_{n=1}^{N} \log p(y_n; \Theta_{LM}) \right\} \tag{1.5}$$

$$\hat{\Theta}_{TM} = \arg\max_{\Theta_{TM}} \left\{ \sum_{n=1}^{N} \log p(x_n|y_n; \Theta_{TM}) \right\} \tag{1.6}$$

In the following sections, the main models classically developed in the literature for estimating the language and translation probability distributions will be reviewed.

1.1. Language Modelling

As seen above, the goal of a language model is to estimate the probability distribution $Pr(y_1^I)$, that is, to capture the linguistic regularities of a natural language which determine how well the sentence y_1^I is formed, according to the language $\mathcal{E}$.

Let the sentence y_1^I be composed of the words $y_1 \dots y_I$. Applying the chain rule, the probability of such sentence, $Pr(y_1^I)$, can be rewritten as:

$$Pr(y_1^I) = Pr(y_1) \cdot Pr(y_2|y_1) \cdot \ldots \cdot Pr(y_I|y_1, \ldots, y_{I-1}) = \prod_{i=1}^{I} Pr(y_i|y_1^{i-1}) \tag{1.7}$$

Computing this expression is intractable, hence, some simplifications or assumptions over it have to be considered.

The classic approach to language modelling, the so-called n-gram language models make Markov assumptions, considering that a word y_i depends only on its $n-1$ preceding words. The value of n defines the *order* of the n-gram. Considering this assumption, Equation 1.7 is rewritten as:

$$Pr(y_1^I) \approx \prod_{i=1}^{I} p(y_i|y_{i-n+1}^{i-1}) \tag{1.8}$$

The estimation of the probability $p(y_i|y_{i-n+1}^{i-1})$, is performed through a ML estimator, counting and normalizing:

$$p(y_i|y_{i-n+1}^{i-1}) = \frac{c(y_{i-n+1}^{i-1})}{\sum_{y_i} c(y_{i-n+1}^{i})} \tag{1.9}$$

where $c(y_{i-n+1}^{i-1})$ corresponds to the number of occurrences of the n-gram y_{i-n+1}^{i-1} in the training corpus.

1.1.1. Smoothing of n-gram Language Models

Since the estimation of n-gram language models is based on counts, if in test phase an unseen word is found, its counts would be zero and the model would output the null probability for the full sequence. This behaviour is obviously undesired. For dealing with this problem, the development of the so-called *smoothing* techniques is required. This term describes techniques for adjusting the maximum likelihood estimate of probabilities to produce more accurate and softer probabilities. These techniques tend to produce a more uniform distribution, increasing low probabilities (zero or almost equal to zero) and decreasing high probabilities (one or almost equal to one). Besides preventing null probabilities, the smoothing methods also improve the global performance of the model (Chen and Goodman, 1998).

In SMT, the most widely used smoothing method is the so-called Kneser-Ney discount (Kneser and Ney, 1995), which is an extension of the *absolute discounting* method. This latter method interpolates high-order and lower-order n-gram models. The high-order distribution is created by reserving a probability mass from each non-zero count, that is, subtracting to each count a fixed discount $D < 1$. Lower-order distributions are smoothed versions of the lower-order maximum likelihood distributions. But in the case that seldom or no counts are present in the higher-order distribution, the lower-order distribution represents an important factor of the global model. Thus, if in the training data are few counts, an important part of the power of the high-order model is on the lower-order models.

The goal of the Kneser-Ney discount is to optimize the lower-order distributions in order to achieve a good performance in this situation. They proposed the following smoothing equation:

$$p_{KN}(y_i|y_{i-n+1}^{i-1}) = \begin{cases} \frac{max\{c(y_{i-n+1}^{i})-D,0\}}{\sum_{y_i} c(y_{i-n+1}^{i})} & \text{if } c(y_{i-n+1}^{i}) > 0 \\ \\ \gamma(y_{i-n+1}^{i-1}) \cdot p_{KN}(y_i|y_{i-n+2}^{i-1}), & \text{if } c(y_{i-n+1}^{i}) = 0 \end{cases} \tag{1.10}$$

where $\gamma(y_{i-n+1}^{i-1})$ is chosen to make the sum of the probability for all possible event equal to 1. Therefore, the lower-order distribution is interpolated with all words, not only with the words with zero counts in the higher order distribution.

1.2. Translation Modelling

Translation modelling tackles the probability $Pr(x_1^J|y_1^I)$ from Equation 1.2. A good translation model should assign high probability to strings x_1^I which are good translations of f_1^J, even if they are incorrectly formed; checking the correctness of a sentence is a task for the language model.

1.2.1. Single-Word Alignment Models: IBM Models

The first approach to translation modelling in modern SMT were the single-word alignment models. They relied on the concept of *alignment*. According to Brown et al. (1993), an alignment between two strings, x_1^J in the source language $\mathcal{F}$ and y_1^I in the target language $\mathcal{E}$, is a correspondence which indicates, for each word of the string x_1^J, the word from the y_1^I string from which it arose:

$$a \subseteq 1...J \times 0...I \tag{1.11}$$

where $a_j = i$ if the j-th source position (x_j) is aligned with the i-th target position (y_i). In addition, an artificial position is generated and placed in index 0 (for convention). It means that the word from x_1^J which is aligned with this null position, is aligned with no word from sentence y_1^I.

Let $\mathcal{A}(x_1^J, y_1^I)$ be the set of all possible alignments between two sentences x_1^J, y_1^I; and let $Pr(x_1^J, a_1^J|y_1^I)$ be the probability of, given the alignment hidden variable a_1^J, to translate the sentence x_1^J into y_1^I. The translation probability can be rewritten as:

$$Pr(x_1^J|y_1^I) = \sum_{a_1^J \in \mathcal{A}(x_1^J, y_1^I)} Pr(x_1^J, a_1^J|y_1^I) \tag{1.12}$$

Since $|y| = I$ and $|x| = J$, $I \times J$ possible connections can be formed between the strings. Hence, $\mathcal{A}(x_1^J, y_1^I)$ contains $2^{I \times J}$ alignments, which may represent a computational challenge.

IBM Models

In 1993, IBM researchers developed the so-called *IBM models* (Brown et al., 1993). They defined five models, based on the concept of alignment between words. The starting point is the following derivation of Equation 1.12:

$$Pr(x_1^J, a_1^J|y_1^I) = Pr(J|y_1^I) \prod_{j=1}^{J} Pr(a_j|x_1^{j-1}, a_1^{j-1}, y_1^I) \cdot Pr(x_j|x_1^{j-1}, a_1^j, y_1^I) \tag{1.13}$$

This equation states that, regardless of the form of $Pr(x_1^J, a_1^J|y_1^I)$, it can always be decomposed in a product of terms in this way. This establishes a sequential way of building a translation. When a string x_1^J is generated, the following steps are followed:

1. The length of x_1^J is chosen, according to y_1^I. This is modelled by the *length distribution* $Pr(J|y_1^I)$.

2. It is decided where to connect the j-th position of x, (a_j), given the knowledge from the built-so-far sentence and alignments from x (x_1^{j-1}, a_1^{j-1}) and the input string (y_1^I). The *alignment distribution* $Pr(a_j|x_1^{j-1}, a_1^{j-1}, y_1^I)$ copes with this task.

3. Finally, the identity of the j-th word in the generated string (x_j) is determined, given the knowledge provided by the input string y_1^I, the built-so-far target sentence x_1^{j-1} and the current alignments, including the one generated in the last step (a_1^j). This is achieved by the *lexical distribution* $Pr(x_j|x_1^{j-1}, a_1^j, y_1^I)$.

This formulation enforces to iterate over points 2 and 3. As the iteration process is carried on, decisions which build up the target string are taken, based on the previous knowledge from the string. In all five models, points 1 and 3 are common. Length distribution is assumed to be a Gaussian distribution, $Pr(J|y_1^I) \approx \mathcal{N}(J|I)$. Lexical probability is approximated by a statistical dictionary of words, $Pr(x_j|x_1^{j-1}, a_1^j, y_1^I) \approx l(x_j|y_{a_j})$. The models variate the assumptions made in step 2, specifically about the alignment distribution $Pr(a_j|x_1^{j-1}, a_1^{j-1}, y_1^I)$. In short, these differences are:

- **Model 1**: The alignment distribution is meant to be uniform.
- **Model 2**: The alignment distribution $Pr(a_j|x_1^{j-1}, a_1^{j-1}, y_1^I)$ is estimated by a zero-order model $t(a_j|i, J, I)$ which defines dependencies between absolute word positions of target and source sentences.
- **Model 3**: A *fertility model* $n(\phi|y)$ is included. It represents, for each target word y, the probability that y generates ϕ source words. The choice of ϕ depends only on y. The distribution $Pr(a_j|x_1^{j-1}, a_1^{j-1}, y_1^I)$ is approximated by a zero-order model, the so-called *distortion model*, $d(i|a_j, J, I)$; which models the dependencies between absolute word positions of source and target sentences.
- **Model 4** and **Model 5**: A first-order distortion model with dependencies between relative word positions of source and target sentences is used.

Models 3 and 4 are *deficient*. Deficiency is the property of a model of not concentrating all of its probability on the events of interest, but wasting its probability on the so-called generalized strings (strings with alignment positions with many words and positions with none). Model 5 consists in a reformulation of Model 4 which avoids deficiency. Nevertheless, Model 5 has a significantly larger number of parameters than Model 4, which may prevent its practical usage. Deficiency can be seen as the price paid for having computationally tractable models.

In addition to the IBM models, other word-alignment models can be defined. Especially interesting are those based on homogeneous hidden Markov models (HMM). The assumptions taken in this case are similar those from Model 2, but in the HMM approach, a first-order model is used. The alignment probability is then computed as:

$$Pr(a_j|x_1^{j-1}, a_1^{j-1}, y_1^I) \approx t(a_j|a_{j-1}, J, I) \tag{1.14}$$

Hence, Equation 1.12 can be formulated as a HMM:

$$Pr(x_1^J|y_1^I) \approx \mathcal{N}(J|I) \sum_a \prod_j^J t(a_j|a_{j-1}, J, I) l(x_j|y_{a_j}) \tag{1.15}$$

where the alignment model $t(a_j|a_{j-1}, J, I)$ would be the transition probability of the HMM and the statistical dictionary $l(x_j|y_{a_j})$ would be the emission probability.

The parameters of each model are estimated by means of (an approximated version of) the *Expectation-Maximization* algorithm (EM) (Dempster et al., 1977). The approximation is necessary since there is not a closed formula for applying an iterative version of the algorithm. Hence, only brute force can be applied.

The great disadvantage of these models is that they, by ignoring alignments larger than one word, miss the contextual information. For tackling this problem, a new family of models was developed: The so-called *multi-word alignment models*, which were able to capture contextual information.

1.2.2. Phrase-Based Models

In order to solve the inability of single-word alignment models for capturing contextual information beyond one word, multi-word alignment models were proposed. In this approach, alignments are formed between groups of words from source and target sentences, instead of just between single words.

The basic concept behind a multi-word alignment is a *phrase*. A phrase is defined as a set of one or more consecutive words of the source or the target sentences. Given a sentence x_1^J, an unspecified phrase from it is denoted by: $\widetilde{x}$.

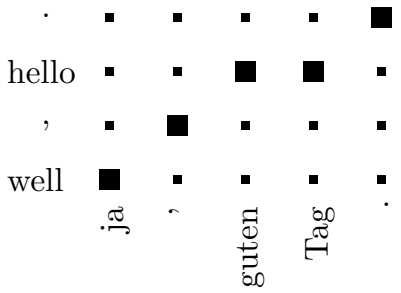
Phrase-based (PB) models make use of the so-called *statistical dictionaries of phrase pairs* (Zens et al., 2002), which consist in sets of bilingual phrases. A bilingual phrase is a pair of m source words and n target words. If they are extracted from a bilingual corpus, the following constraints must be satisfied:

1. The words in the phrase are consecutive.
2. They are consistent with the word alignment matrix: The m source words must be aligned only to the n target words and vice versa.

Figure 1.1 shows an example of word aligned pair and the set of bilingual phrases generated from it.

Under the phrase-based framework, the translation of a source sentence x_1^J to a target sentence y_1^I is computed following the steps:

1. Segmentation of the source sentence into K phrases $x_1^I \equiv \widetilde{x}_1^K$. This introduces the hidden variable μ.
2. Translation of each source phrase into target phrases. The segmentation of the target phrase introduces another hidden variable, γ.
3. Reordering of the target phrases in order to generate the target sentence $y_1^I \equiv \widetilde{y}_1^K$. Similarly to word-based translation, an alignment variable α must be defined. In this case, the alignments are produced between phrases.



(a) Word aligned sentence pair.

Source phrase	Target phrase
ja	well
ja,	well,
ja, guten Tag	well, hello
ja, guten Tag.	well, hello.
,	,
, guten Tag	, hello
, guten Tag.	, hello.
guten Tag	hello
guten Tag.	hello.
.	.

(b) Bilingual phrases.

Figure 1.1: Aligned sentence pair and set of bilingual phrases extracted from the alignments. Example borrowed from Zens et al. (2002).

Hence, three additional hidden variables must be included in order to work with bilingual phrases. μ and γ constitute the so-called *bisegmentation* of source and target sentences. A bisegmentation of length K of a sentence pair (x_1^J, y_1^I), represents a segmentation of the sentence pair into K phrases: $(\widetilde{x}_1^K; \widetilde{y}_1^K)(1 \leq K \leq min(I, J))$. A bisegmentation is a phrase-level alignment of a sentence pair.

Phrase-based models are inspired in word-level HMM (Equation 1.15). Including all hidden variables into the probability computation, the following expression is reached:

$$Pr(x_1^J|y_1^I) \approx \mathcal{N}(J|I) \sum_K \sum_{\mu_1^K} \sum_{\alpha_1^K} \sum_{\gamma_1^K} \prod_{k=1}^K q(\alpha_k|\alpha_{k-1}, K) p(x_{\mu_{k-1}+1}^{\mu_k}|y_{\gamma_{\alpha_{k-1}+1}}^{\gamma_{\alpha_k}}) \tag{1.16}$$

In this approach, instead of a statistical dictionary, phrase tables $p(\widetilde{x}|\widetilde{y})$ are used; and the first-order alignment model $q(\alpha_k|\alpha_{k-1}, K)$ works at phrase level instead of at word level.

If monotonic alignments are assumed ($\alpha_k = k$), alignments would be created sequentially:

$$Pr(x_1^J|y_1^I) \approx \mathcal{N}(J|I) \sum_K \sum_{\mu_1^K} \sum_{\gamma_1^K} \prod_{k=1}^K p(x_{\mu_{k-1}+1}^{\mu_k}|y_{\gamma_{k-1}+1}^{\gamma_k}) \tag{1.17}$$

The monotonicity restriction allows to a faster probability computation. But it also should be noted that it can lead to many translation errors in languages with different phrase order, for example, English-German or English-Japanese.

Since we are interested only in the sentence with highest probability, the search problem is usually tackled via the maximum approximation:

$$Pr(x_1^J|y_1^I) \approx \mathcal{N}(J|I) \max_K \max_{\mu_1^K} \max_{\alpha_1^K} \max_{\gamma_1^K} \prod_{k=1}^K q(\alpha_k|\alpha_{k-1}, K) p(x_{\mu_{k-1}+1}^{\mu_k}|y_{\gamma_{\alpha_{k-1}+1}}^{\gamma_{\alpha_k}}) \tag{1.18}$$

1.3. The SMT Log-linear Model

The first SMT systems tackled the translation process through generative models (Equation 1.2). More recent approaches replaced such models by *discriminative models*, which directly model the posterior probability $Pr(y_1^I|x_1^J)$ (Equation 1.1). Most current SMT systems rely on these approach, more precisely, on the so-called *log-linear model*. This model consists in a set of feature functions $h_m(x_1^J.y_1^J)$, each one with a corresponding weight λ_m:

$$Pr(y_1^I|x_1^J) \approx \frac{\exp\left(\sum_{m=1}^{M} \lambda_m h_m(x_1^J, y_1^I)\right)}{\sum_{y_1'^{I'}} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(x_1^J, y_1'^{I'})\right)} \tag{1.19}$$

Since the denominator of this expression is independent from the target sentence y_1^I, it can be omitted in the search process:

$$\hat{y_1^I} = \arg\max_{I, y_1^I} \sum_{m=1}^{M} \lambda_m h_m(x_1^J, y_1^I) \tag{1.20}$$

Among the functions h_m included in the SMT systems more frequently, the following features can be found:

1. The target language model $p(y_1^I)$.
2. The phrase-based model and inverse phrase-based model, $p(e_1^I|f_1^J)$ and $p(f_1^J|e_1^I)$. They are extracted from the phrase tables (Equation 1.16).
3. A reordering model $q(a_j|a_{j-1})$.
4. A target word penalty (ωI) and phrase penalty (ρK). Due to the nature of statistical models, they tend to obtain short sentences and few number of phrases. These penalties aim to overcome this issue, obtaining a more adequate length of the output sentence and number of phrases.

It should be noted that the log-linear framework accepts the inclusion of more models. For instance, the toolkit Moses (see Section 5.1.1) include lexicalized reordering models or additional neural models (Baltescu et al., 2014; Devlin et al., 2014).

The learning of the weights λ_m usually follows a criterion of minimization of an error function, typically $(1 - BLEU)$ (see Section 1.4). The weights are fixed in order to maximize the performance over a validation set. Different optimization algorithms are proposed in the literature, for instance, MERT (Och, 2003), downhill simplex (Melder and Nead, 1965), MIRA (Hasler et al., 2011) or PRO (Hopkins and May, 2011).

1.4. Assessment

The evaluation of automatic translations is still an open problem. Evaluation metrics can be categorized into two main categories: Human evaluation and automatic evaluation. With respect to the former, human evaluation generally produces high quality metrics. But on the other hand, it is prohibitively slow

and expensive: First, human linguists must define criteria for evaluating adequacy and fluency. Next, human judges must analyse system translations and evaluate them in terms of such criteria. Besides its cost, human evaluation must deal with the ambiguity inherent to the human being: Two judges can diverge with respect to the same translation and qualify it differently. This makes the use of human evaluation unrealistic.

Hence, it is necessary to develop automatic metrics, which try to follow a similarity with human evaluation. Automatic metrics must be objective, informative, efficient and cheap. These methods usually compare the output of the system with high-quality human translations, called references. In this work, the translation performance has been evaluated using the most extended metric in the research community: *BLEU* (Papineni et al., 2002).

BLEU

BLEU (BiLingual Evaluation Understudy) aims to model the correspondence between the translation generated by the MT system and the one produced by a human translator. The metric is based on the so-called *n-gram precision*. The n-gram precision counts the number of candidate words from the system translation which appear in the reference sentence, and divides this count by the total number of words of the translation from the system. Since a system can generate too many suitable n-grams, it is necessary to restrict such counts. This restriction produces the *modified n-gram precision*, where the counts of an n-gram $\mathtt{w}$ are clipped following:

$$C_{clip}(\mathtt{w}) = \min(c(\mathtt{w}), Max\_Ref\_C(\mathtt{w}))$$

where $c(\mathtt{w})$ are the counts of $\mathtt{w}$ in the hypothesis sentence and $Max\_Ref\_C(\mathtt{w})$ are the maximum counts of $\mathtt{w}$ in a single reference sentence in the reference corpus.

The basic computation unit for BLEU score is a sentence and the computation is extended to the whole test corpus. Finally, an unique modified precision score for the entire corpus is generated by 1. computing the n-gram matches sentence by sentence and 2. adding the clipped n-gram counts for all these candidate sentences. Finally, this result is normalized by the number of candidate n-grams in the test corpus:

$$p_n = \frac{\sum_{\mathcal{C}\in\{Candidates\}} \sum_{\mathtt{w}\in\mathcal{C}} C_{clip}(\mathtt{w})}{\sum_{\mathcal{C}'\in\{Candidates\}} \sum_{\mathtt{w}'\in\mathcal{C}'} C_{clip}(\mathtt{w}')} \tag{1.21}$$

Moreover, a translated sentence should be neither excessively long or short. This must be also reflected in the metric: System translations longer than references are naturally penalized by the modified n-gram precision. For handling shorter translations, it is introduced the so-called *brevity penalty (BP)*, which penalizes short translations following:

$$BP = \begin{cases} 1, & \text{if } t > \mathrm{r} \\ \\ e^{1-\frac{r}{t}}, & \text{if } t \leq \mathrm{r} \end{cases} \tag{1.22}$$

where t is the length of the system hypothesis and r is the length of the reference sentence.

BLEU score uses a weighted geometric mean of the n-grams for combining them. Each n-gram has a weight w_n such that $\sum_{n=1}^{N} w_n = 1$. Typically this weight is set to $w_n = \frac{1}{N}$. The maximum order of the n-grams considered by the metric is generally fixed to $N = 4$. This value was set according to experimental results (Papineni et al., 2002). Hence, the final BLEU score is computed as:

$$BLEU = BP \cdot exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{1.23}$$

Perplexity

For evaluating a language model, the metric must measure how similar the estimated distribution of the model, determined by the parameters Θ_{LM}, with respect to the real (and unknown) probability distribution $Pr(y_1^I)$. The most extended metric in evaluation of statistical language models is the *Perplexity*. It is defined over the empirical estimation of the *cross entropy* of those distributions. Let $\mathcal{T} = \{y_1, y_2, ..., y_N\}$ be a set of test samples and Θ_{LM} the set of language model parameters. The cross entropy is defined as:

$$H(Pr; p(\cdot|\Theta_{LM})) = -\sum_{i=1}^{N} Pr(y_i) \cdot \log_2 p(y_i|\Theta_{LM}) \tag{1.24}$$

Perplexity is then computed as:

$$PPL(Pr; p(\cdot|\Theta_{LM})) = 2^{H(Pr; p(\cdot|\Theta_{LM}))} \tag{1.25}$$

The perplexity of a language model can be seen as the ability of the model of predicting a sample. It also can be understood as the geometric average of the branching factor of the given language with respect the model. It measures either the model performance and the complexity of the task.

Other Metrics

METEOR (Denkowski and Lavie, 2014) is a specific language metric, which makes use of the available electronic language resources (like WordNet or Snowball Stemmer) for the evaluation. The scoring is based on alignments between a hypothesis and a reference sentence. Over these alignments, four matches are considered: Both alignments are equal, the stem of both alignments is the same, both alignments are synonyms and both alignments are considered a paraphrase. The harmonic mean of the precision and recall of each matcher is computed. The final score provided by meteor is this mean, penalized by differences in word order. According to their authors, METEOR outperforms the BLEU metric with respect to correlation with human measures, making it more similar to human judgement. The main drawback of this metric is that, since it makes use of specific language resources, its usage is limited to few pairs of languages. In this work, METEOR was used to compute alignments between different system hypothesis, in a combination system (Section 3.2.1).

Apart from these, many other techniques for assessing translations have been proposed in the literature, like word error rate (WER), translation edit rate (TER) (Snover et al., 2006) or NIST (Doddington, 2002).

Chapter 2

Neural Language Models for Machine Translation

The classic models for machine translation and language modelling (those seen in the previous chapter) rely on a discrete representation space. Thanks to the smoothing techniques (e.g. Chen and Goodman (1998)), the n-gram language models tackle the data sparseness problem, being capable of obtaining predictions for non-seen events. This leads to simple, robust and fast models, which may be trained over huge amounts of data. But of course, this simplicity entails weaknesses: Since words are merely treated as indices in a vector, concepts such as similarity or semantic relationships between words are missing.

Moreover, the n-gram model, only considers few context words: Typically, the order of n-gram models typically ranges from 2 to 5, therefore the model takes from 1 to 4 context words and long-term relationships are lost (Bengio et al., 2003). Increasing the order of the n-gram further than 5 is generally ineffectual, due to data scarcity: High-order n-grams are less frequent than lower-order ones. When considering high-order n-grams, most times is an unseen n-gram hence, the model backs off to lower order n-grams, making the high-order ineffective. In order to address these issues, more complex approaches for language modelling were developed.

In the last years, one of the most explored and successful approaches rely on a distributed representation of words: A real-valued, dense and low-dimensional representation in a continuous space. In these models, probability estimation is carried out in this continuous space, typically by means of a neural network. Furthermore, given the nature of continuous models, the learned function is inherently smoothed.

In this chapter, a model belonging to this family is proposed. It aims to overcome the aforementioned drawbacks of n-gram language models: First, by projecting the words into a continuous space, the model profits from richer representations of words. Second, by using a bidirectional recurrent neural network, the context of a word is determined not only by its preceding words, but also by its following words.

2.1. Recurrent Neural Networks

A recurrent neural network (RNN) is a class of artificial neural network where the connections between units form a directed cycle. This creates an internal state of the network which allows it to model a discrete-temporal behaviour. Given an input sequence of vectors $\mathbf{x}_1^T = \mathbf{x}_1, \ldots, \mathbf{x}_T$, a RNN produces an output sequence $\mathbf{y}_1^T = \mathbf{y}_1, \ldots, \mathbf{y}_T$, computed as:

$$\mathbf{h}_t = f_h(\mathbf{x}_t, \mathbf{h}_{t-1}) \tag{2.1}$$

$$\mathbf{y}_t = f_o(\mathbf{h}_t) \tag{2.2}$$

where $\mathbf{h}_t$ is the hidden state of the network at time-step t, f_h is a hidden state function and f_o is an output function.

Different choices of $\mathbf{h}$, f_h and f_o lead to the different RNN architectures proposed in literature, like Jordan networks (Jordan, 1990) or Elman networks (Elman, 1990). Figure 2.1 shows the latter architecture, which consists in an input layer, a self-connected hidden layer and an output layer. Here, previous functions are defined as:

$$\mathbf{h}_t = f_h(\mathbf{x}_t, \mathbf{h}_{t-1}) = \phi(\mathbf{W}^\top h_{t-1} + \mathbf{U}^\top x_t) \tag{2.3}$$

$$\mathbf{y}_t = f_o(\mathbf{h}_t) = \sigma(\mathbf{V}^\top h_t) \tag{2.4}$$

where $\mathbf{W}, \mathbf{U}, \mathbf{V}$ are the recurrent, input and output weight matrices respectively, ϕ is a non-linear activation function, commonly sigmoid or hyperbolic tangent, and σ is the output activation function. An usual choice for σ is the *softmax* function (Equation 2.10).

These three weight matrices form the set of parameters of the model and are typically estimated by a stochastic gradient descend method, using the back-propagation through time algorithm (Werbos, 1990), for minimizing a cost function under some optimality criterion, typically cross-entropy between the output of the system and the training data probability distribution.

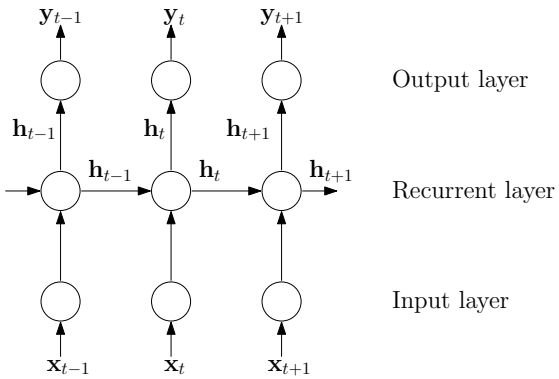


Figure 2.1: Elman architecture of a RNN, unfolded three steps in time.

2.1.1. Bidirectional Neural Networks

A drawback of regular RNNs is that the input sequence is only scanned in one direction, normally from past to future. In order to capture both past and

future context, bidirectional recurrent neural networks (BRNN) were proposed by Schuster and Paliwal (1997). The main idea is to have two independent recurrent layers: One layer process the input sequence in forward time direction (from 1 to T), while the other layer process the input sequence reversed in time (from T to 1). Since hidden layers have no interaction between them, bidirectional RNNs can be trained using the same algorithms as those used for unidirectional RNNs. Following prior notation, a bidirectional RNN is defined as:

$$\mathbf{h}_t^{\mathrm{f}} = f_h(\mathbf{x}_t, \mathbf{h}_{t-1}^{\mathrm{f}}) = \phi(\mathbf{W}_{\mathrm{f}}^\top \mathbf{h}_{t-1}^{\mathrm{f}} + \mathbf{U}^\top \mathbf{x}_t) \tag{2.5}$$

$$\mathbf{h}_t^{\mathrm{b}} = f_h(\mathbf{x}_t, \mathbf{h}_{t+1}^{\mathrm{b}}) = \phi(\mathbf{W}_{\mathrm{b}}^\top \mathbf{h}_{t+1}^{\mathrm{b}} + \mathbf{U}^\top \mathbf{x}_t) \tag{2.6}$$

$$\mathbf{y}_t = f_o(\mathbf{h}_t^{\mathrm{f}}, \mathbf{h}_t^{\mathrm{b}}) = \sigma(\mathbf{V}^\top (\mathbf{h}_t^{\mathrm{f}} + \mathbf{h}_t^{\mathrm{b}})) \tag{2.7}$$

where $\mathbf{h}_t^{\mathrm{f}}$ is the forward layer and $\mathbf{h}_t^{\mathrm{b}}$ is the backward layer.

As before, $\mathbf{U}$ and $\mathbf{V}$ are the matrices associated to the input and output layers, while $\mathbf{W}_{\mathrm{f}}$ and $\mathbf{W}_{\mathrm{b}}$ are the weight matrices of the forward and backward layers. The output is a combination produced by the output function f_o of both backward and forward layers. A scheme of this architecture is shown in Figure 2.2.

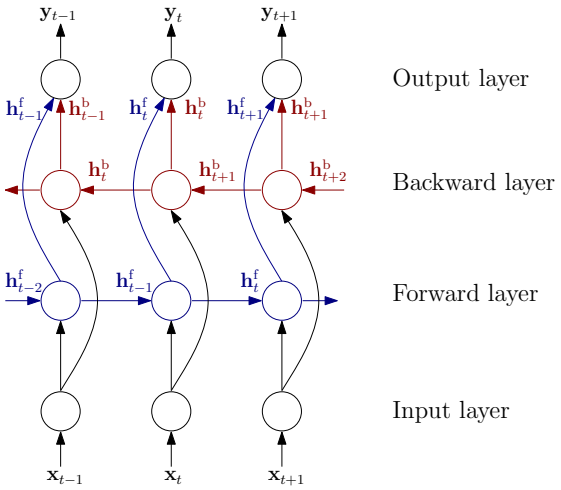


Figure 2.2: Bidirectional architecture of a RNN, unfolded in time. For the sake of clarity, forward components are blue-coloured and backward components are red-coloured.

2.2. A Bidirectional Recurrent Language Model

As stated in Section 1.1, the task of statistical language modelling consists in estimating the probability distribution over a sequence of words $Pr(y_1^I), y_i \in \mathcal{E}$, where $\mathcal{E}$ is a large but finite vocabulary set.

2.2.1. Neural Language Modelling: Related Work

One of the first neural language models was proposed by Bengio et al. (2003). They defined a function f over a subsequence of n words $f(y_i, ..., y_{i-n+1}) \approx Pr(y_1^I)$, which had the goal of producing a low perplexity. Note that the model maintained the same assumption than an n-gram approach: To discard dependencies further than n words.

The key contribution of this work was the decomposition of $f(y_i, ..., y_{i-n+1})$ in two parts:

- **Word embedding**: Consisted in a mapping for each word index of the vocabulary to a continuous space. A word embedding is a function $c : \mathbb{N} \rightarrow \mathbb{R}^m$ from any element $y \in \mathcal{E}$ to a real vector $c(y) \in \mathbb{R}^m$, being m the size of the vector.
- **Word probability function**: An estimation of the posterior probability of the words, performed in the continuous space. This is achieved by means of a feed-forward or recurrent neural network. The output of this neural network is a vector which i-th element estimates the probability $Pr(y_i = i|y_1^{i-1})$.

These both parts were trained jointly, with the maximization of the log-likelihood of the training data as objective function. Thus, the model had more power of generalization than classic n-grams and was able to deal better with the curse of dimensionality of the data.

From this work, many other researchers followed these ideas. Bengio et al. (2003) and Schwenk (2013) performed the probability estimation through a feedforward neural network. Mikolov (2012) used a recurrent neural network (RNN) for this purpose. In his model, the projection layer was removed, words were mapped directly to the hidden layer of an Elman network. Sundermeyer et al. (2012) combined both models, having a projection layer connected to a recurrent layer with long short-term memory (LSTM) units. Pascanu et al. (2014) extended the recurrent architecture, which led to *deep RNNs*, and they were applied to language modelling with satisfactory results.

In the field of SMT, neural language models also have recent applications: Baltescu et al. (2014) coupled a feedforward neural language model into the SMT decoder. Wang et al. (2014) approximated a neural language model with an n-gram language model, according to bilingual information extracted from the phrase table. Devlin et al. (2014) extended the original neural language model from Bengio et al. (2003), and developed a neural translation model. This model could be integrated into a hierarchical decoder and offered impressive results in terms of translation performance.

2.2.2. A Bidirectional Recurrent Language Model

In the RNN framework, information about the history $(y_1, \ldots, y_{i-1})$ is represented in the hidden recurrent layer $(\mathbf{h}_i)$. Thus, sequence probability is rewritten as:

$$p(y_1, \ldots, y_I) = \prod_{i=1}^{I} p(y_i|\mathbf{h}_i) \tag{2.8}$$

If BRNN are considered, the probability is conditioned by both forward and backward layers ($\mathbf{h}_i^{\mathrm{f}}, \mathbf{h}_i^{\mathrm{b}}$, respectively):

$$p(y_1, \ldots, y_I) = \prod_{i=1}^{I} p(y_i | \mathbf{h}_i^{\mathrm{f}}, \mathbf{h}_i^{\mathrm{b}}) \tag{2.9}$$

In the proposed language model architecture, input words are one-hot vectors, binary vectors with all elements set to 0 except the index which represents the input word, which is set to 1. Those vectors are projected into the continuous space through a projection layer and then fed to the BRNN. As architectural choices of the network, the hidden function (f_h) is the sigmoid function. The output function (f_o) is modelled through a 2-layer perceptron. The activation function of its first layer is the sigmoid function. This first layer is fully connected to the output layer, which makes use of the softmax cost function in order to obtain correct output probabilities:

$$\sigma(z_k) = \frac{\exp(z_k)}{\sum_{k'=1}^{\mathcal{K}} \exp(z_{k'})} \tag{2.10}$$

where z_k is the k-th output unit. Each output unit represents a word in the vocabulary, hence, the output layer is vocabulary-sized.

At test time, the probability of a sentence is normalized with respect to the length of the sentence, in order to prevent benefits to short sentences (Graves, 2013). Since using the full vocabulary is computationally unaffordable, a shortlist is used: Only the most $\mathcal{K}$ frequent words are taken into account. The rest are mapped to a special token `<UNK>`. Figure 2.3 shows a scheme of the model.

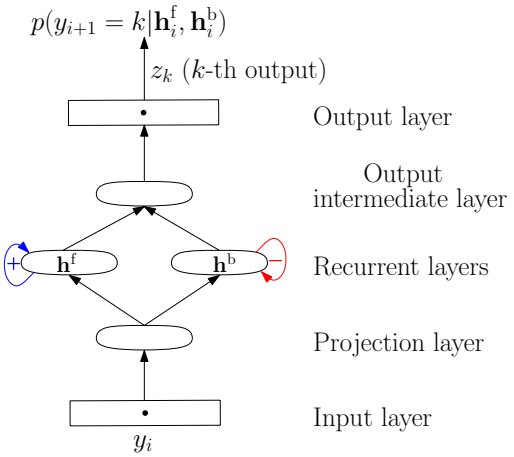


Figure 2.3: Architecture of the bidirectional language model, similarly depicted as in Sundermeyer et al. (2014). Input and output layers have the size of the vocabulary.

Chapter 3

Neural Machine Translation

Despite the idea of using neural networks for machine translation was conceived long ago (Castaño et al., 1997), in the last years novel and encouraging approaches for neural machine translation (NMT) were developed. NMT employs only a large neural network for performing translations: The inputs of the network are the source sentences and its outputs are the translations. In the last year, two similar models for NMT were independently proposed by Sutskever et al. (2014) and Cho et al. (2014). Both works rely on the encoder-decoder framework: The encoder reads a source sentence and projects it into a continuous, fixed-length representation, while the decoder takes this representation and outputs a sentence in the target language, which corresponds to the translation of the source sentence, as shown in Figure 3.1.

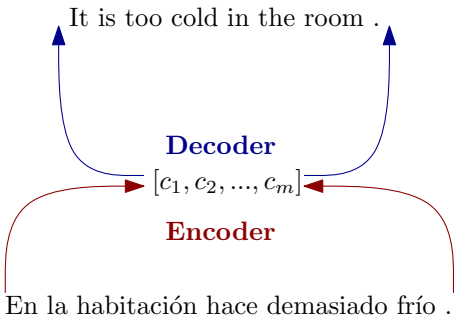


Figure 3.1: Main idea of NMT: The encoder represents the source sentence in Spanish as a distributed vector (**c**). The decoder takes this representation and projects it into the corresponding sentence in the target language (English).

3.1. Neural Machine Translation

The differences between the models proposed in the literature mainly depend on specific architectural choices made upon this framework. Sutskever et al. (2014) used a deep LSTM neural network, while Cho et al. (2014) used a LSTM-like unit in order to integrate the neural network model into a phrase-based system. Bahdanau et al. (2014) extended this latter idea, allowing the neural

network to learn the alignments and decode the translations by itself. In the following sections, this model is described.

In the encoder-decoder framework, a variable-length word input sequence is mapped into a fixed-size vector. This vector is later decoded this vector into anther variable-length word sequence, in the target language. Given a set of training pairs, $\mathcal{X} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, the training objective of the model is to maximize the conditional log-likelihood (Equation 3.1) over the training set:

$$\max_{\Theta} \frac{1}{N} \sum_{n=1}^{N} \log p_{\Theta}(y_n|x_n) \tag{3.1}$$

where $x_n = x_1, \ldots, x_J$ is an input sentence of length J, $y_n = y_1, \ldots, y_I$ is an output sentence of length I and Θ is the set of parameters. Note that J may differ from I.

3.1.1. Encoder

The encoder is a RNN which reads each input sequence of words $x_1^J = x_1, \ldots, x_J$ and encloses it into a context vector $\mathbf{c}$. After reading each input element x_j, the hidden state of the RNN h_j changes. When the whole input sequence has been read, the hidden state is a compendium of the sequence. The context vector $\mathbf{c}$ is built by applying a non-linear function to the hidden state sequence:

$$\mathbf{h}_j = f(x_j, \mathbf{h}_{j-1}) \tag{3.2}$$

$$\mathbf{c} = q(\{\mathbf{h}_1, \ldots, \mathbf{h}_J\}) \tag{3.3}$$

where $\mathbf{h}_j \in R^m$ is the hidden state at a time j and f and q are non-linear functions.

The architectural choices taken for the encoder consist in a bidirectional RNN (Section 2.1.1), which forward states are fed with the ordered input sequence (from x_1 to x_J) and its backward states read the reverse-ordered input sequence (from x_J to x_1). Thus, the forward layer ($\mathbf{h}_j^{\mathrm{f}}$) computes a sequence of forward hidden states ($\mathbf{h}_1^{\mathrm{f}}, \cdots, \mathbf{h}_J^{\mathrm{f}}$), while the backward layer ($\mathbf{h}_j^{\mathrm{b}}$) computes similarly a sequence of backward hidden states ($\mathbf{h}_J^{\mathrm{b}}, \cdots, \mathbf{h}_1^{\mathrm{b}}$). For each word x_j, an annotation h_j is obtained by concatenating both forward and backward states: $\mathbf{h}_j = \left[\mathbf{h}_j^{\mathrm{f}\top}; \mathbf{h}_j^{\mathrm{b}\top}\right]^{\top}$. Therefore, $\mathbf{h}_j$ contains a representation of the preceding and following words of x_j. Given the RNN nature, the hidden state $\mathbf{h}_j$ of a word x_j, will focus on the surrounding words of x_j.

3.1.2. Decoder

The decoder is trained to generate the output sentence $y_1^I = y_1, ..., y_I$ given the previous predicted words and the context vector $\mathbf{c}$. Applying the chain rule, the following expression is obtained:

$$p(y_1, ..., y_I) = \prod_{i=1}^{I} p(y_i|\{y_1, ..., y_{i-1}\}, \mathbf{c}) \tag{3.4}$$

In order to have tractable models, it is assumed that only exist direct dependencies with the previous generated word (y_{i-1}). In a RNN scope, each conditional probability is modelled as:

$$p(y_i|\{y_1, ..., y_{i-1}\}, \mathbf{c}) = g(y_{i-1}, \mathbf{s}_i, \mathbf{c}) \tag{3.5}$$

where g is a non-linear function and $\mathbf{s}_i$ is the hidden state of the decoder RNN.

Note that, in this approach, each sentence is represented in the fixed-size vector $\mathbf{c}$. If the sentence is long, the context vector will be unable to correctly capture all the information and relationships within the sentence. This phenomenon was noticed by Cho et al. (2014), who observed that the performance of the system was greatly decreased when long sentences were involved. In order to overcome this issue, Bahdanau et al. (2014) employed a different context vector $\mathbf{c}_i$ for each target word y_i, i.e. a *variable-length* context vector. In this way, long sentences will have long sequences of context vectors, being capable of properly representing all the sentence information and relationships.

By defining a different context vector at each time step ($\mathbf{c}_i$), Equation 3.5 is rewritten as:

$$p(y_i|\{y_1, ..., y_{i-1}\}, \mathbf{c}_i) = g(y_{i-1}, \mathbf{s}_i, \mathbf{c}_i) \tag{3.6}$$

The context vector $\mathbf{c}_i$ is computed as a weighted sum of the sequence of annotations outputted by the encoder:

$$\mathbf{c}_i = \sum_{j=1}^{J} \alpha_{ij} \mathbf{h}_j \tag{3.7}$$

where α_{ij} is the weight assigned to each annotation $\mathbf{h}_j$. This weight is computed following the softmax function:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{J} \exp(e_{ik})} \tag{3.8}$$

where $e_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j)$ is a score provided by a soft alignment model, which measures how well the inputs around position j and outputs around position i match. Figure 3.2 shows the architecture of the NMT system.

Unlike the output layer of the neural language models, where the normalization (denominator from Equation 3.8) was done for all the vocabulary, in this case, is performed for all input annotations, which have a tractable size. Hence, the computation of Equation 3.8 is computationally tractable.

Soft Alignment Model

The alignment model is based on the immediately previous state of the decoder RNN, $\mathbf{s}_{i-1}$, and on the annotation for the source sentence $\mathbf{h}_j$. Since it needs to be evaluated $J \times I$ times, it cannot be an expensive model. Because of this, the alignment model consists in a single-layer perceptron. The alignment score is computed following:

$$a(\mathbf{s}_{i-1}, \mathbf{h}_j) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a \mathbf{s}_{i-1} + \mathbf{U}_a \mathbf{h}_j) \tag{3.9}$$

where $\mathbf{W}_a \in \mathbb{R}^{m' \times m}$, $\mathbf{U}_a \in \mathbb{R}^{m' \times 2m}$ and $\mathbf{v}_a \in \mathbb{R}^{m'}$ are the weight matrices. m is the number of units of hidden layer of the encoder RNN and m' is the number of units of the hidden layer of the decoder RNN.

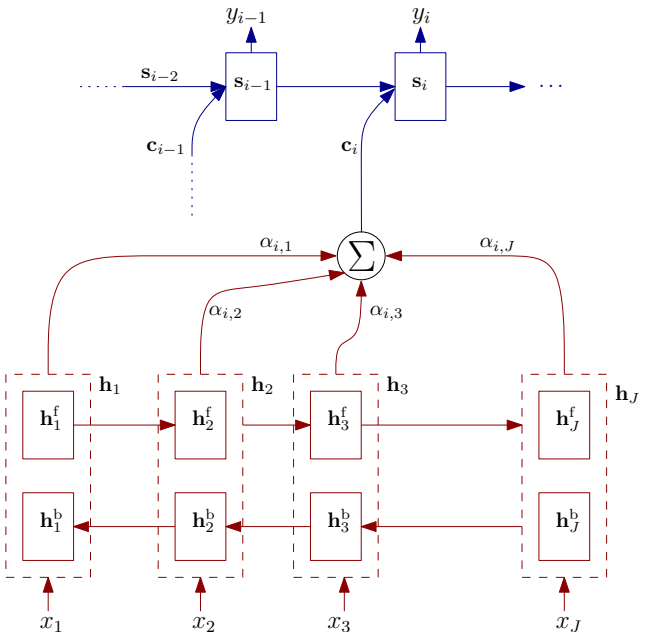


Figure 3.2: Architecture of the neural translation model (Bahdanau et al., 2014). The encoder components are red coloured and the components of the decoder are blue coloured.

3.1.3. Generating translations

Although the means differ, the final goal of a NMT system is the same as the classic one (Equation 1.1). Given a test sentence x_1^J in the source language, its translation $\hat{y}_1^{\hat{I}}$ will be the sentence in the target language with maximum posterior probability. That means, $\hat{y}_1^{\hat{I}}$ is obtained through:

$$\hat{y}_1^{\hat{I}} = \arg\max_{I, y_1^I} \{p(y_1^I | x_1^J)\} \tag{3.10}$$

The search space is all possible sentence in the target language. An optimal resolution of this optimisation is completely unaffordable. Hence, sub-optimal strategies must be used in order to generate translations. One of the most popular search strategies is the so-called *beam search*.

The beam search method maintains at each moment a small number ($\mathcal{B}$) of partial hypotheses. In this case, a partial hypothesis is the prefix of a translation. At each time-step, each partial hypothesis is augmented with all possible word of the target language but, once again, all but the $\mathcal{B}$ hypotheses with most probability are discarded.

The process continues until the end-of-sequence symbol is generated. In this case, the partial hypothesis is considered to be a complete one.

3.2. System Combination

A way for enhancing the quality of machine translation is to combine the outputs of several MT systems. Different MT engines have different strengths and weaknesses. An appropriate combination of them, profiting from the strengths and addressing the weaknesses, can lead to some enhancement over the performance of the single systems. The translation procedure of NMT systems is completely different than the one of PB systems, therefore, it may add information that PB systems are not able to capture. From the other point of view, PB systems may help the NMT system, helping to tackle the rare word problem.

Diverse approaches for combination of systems were proposed in the literature: The first works were focused on combination of automatic speech recognition systems (Fiscus, 1997). González-Rubio et al. (2011) combined N-best lists with a minimum Bayes-risk technique. Heafield et al. (2009) developed a two-stages approach: First, common words and phrases among the hypotheses were identified. Next, the final translation was decoded from the space of combinations. Other family of combination methods rely on exploiting the so-called *confusion networks* (Bangalore et al., 2001; Matusov et al., 2008; Rosti et al., 2012). In this work, for performing the combination, we used a method belonging to this latter category.

3.2.1. Combination through Confusion Networks

In this approach, the different MT systems are treated as black-boxes which output translation hypotheses, that is, only the output hypotheses of the MT systems are considered. The main idea is to collapse all hypotheses into a confusion network (CN). The decoding of the CN will provide the final translation.

A confusion network is a weighted directed graph which represents a finite-state automaton, with initial and final nodes. The edges are labelled with (sequences of) words. A CN has the property that all paths from the initial node to the final node, visit all nodes.

Let M be the number of MT systems to be combined. Therefore, each sentence in the test set generates M translation hypotheses $y_1, \ldots, y_M$. It is possible to collapse those hypotheses, into a single confusion network. For doing this, an alignment between translations must be computed. Then, a translation is chosen as the primary one (y_m) and the rest ($y_n, 1 \leq n \leq M, n \neq m$) are aligned following the structure of the primary. The empty string ϵ is also considered in the construction of the network. Figure 3.3 shows an example of CN.

In this approach arise three key questions:

1. How to align and reorder the words from the different hypotheses?
2. Which hypothesis is considered to be the primary one?
3. How to decode a CN? Additional features should be used?

Since in this work the combination system used was the one developed in Freitag et al. (2014), their particular choices with respect these issues are briefly explained below.

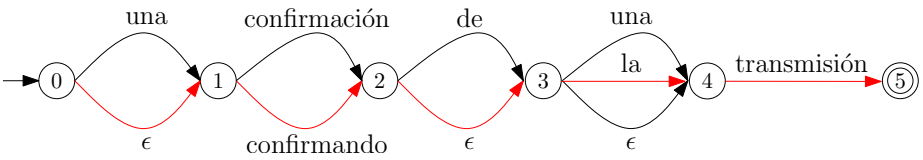


Figure 3.3: Example of a confusion network built from three systems, which hypotheses are: 1. "`una confirmación de la transmisión`", 2. "`confirmando una transmisión`" and 3. "`confirmando transmisión`". The primary hypothesis was the first one. Symbol ϵ denotes the empty word. Let the path coloured in red be the path with highest score. Hence, the combined hypothesis provided by the CN is "`confirmando la transmisión`".

1. Aligning and reordering hypotheses

In order to provide flexibility between close hypothesis (e.g. misspellings, synonyms, etc.), METEOR alignments (see Section 1.4) are used. The confusion network is initialized according the primary translation and the secondary hypotheses are consecutively aligned in the confusion network using METEOR alignments.

2. Selecting the primary hypothesis

The selection of the primary hypothesis is a key point of the combination system, because it decides the ordering of the words of the consensus translation. Since there is no a-priori information about which is the best hypothesis, all hypotheses are considered as primary one time. Hence, M confusion networks are generated for each sentence in the test set. The final confusion network is an union of all these networks, which is jointly decoded.

3. Decoding a confusion network

A weighted combination of features is used for scoring the edges of the network. The weights assigned to the features must be tuned on a development set. This is done by means of the MERT (Och, 2003) method. The baseline system include four types of features:

1. M binary voting features: If the word of an edge is in the hypothesis of the system m ($1 \leq m \leq M$), the feature value is 1; otherwise, is 0.
2. Primary system binary feature: Has a value of 1 for the primary hypothesis and 0 for the rest.
3. Language model: An n-gram language model trained over the hypotheses data.
4. Word penalty: Takes into account the number of generated words. Note that, although the path in a confusion network has always the same length, the number of generated words can vary, due to the ϵ transitions.

Moreover, additional features can be added:

5. Additional language model: Another n-gram language model, trained over additional data (e.g. the monolingual corpus used for training the SMT systems).

6. IBM-1 models: Lexical IBM-1 models, direct and inverse, obtained from training data (see Section 1.2.1).

Decoding a network consists in finding the path which maximizes the value of these features. The consensus translation $\hat{y}$ is generated concatenating the words associated to such path. A complete overview of the system is depicted in Figure 3.4.

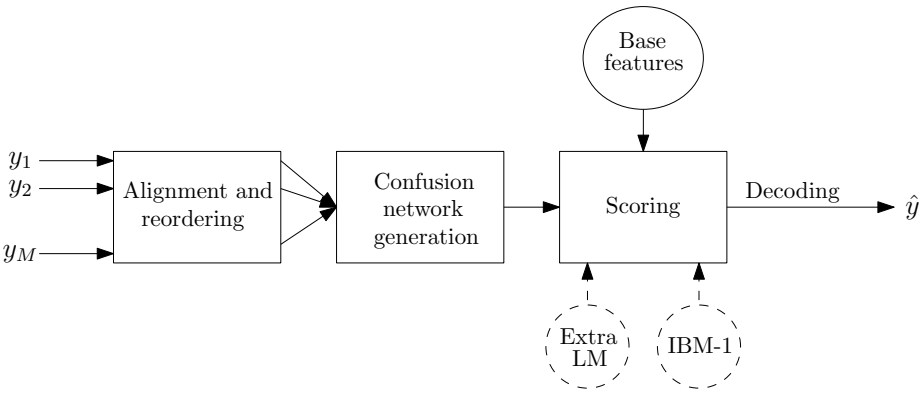


Figure 3.4: Scheme of the combination of systems. Dashed lines refer to optional components to the baseline system. $y_1, \ldots, y_M$ are the translation hypotheses of M MT engines for the same sentence. Such hypotheses are aligned and reordered, building a CN. The edges of the CN are scored according to the features and finally decoded, producing the consensus translation $\hat{y}$.

Chapter 4

Data Selection

Corpora used for training classic SMT systems are typically much larger than the test set. Hence, the training data probably contains noise and sentences which are irrelevant for a specific task. On the other hand, large neural models, as the described before, present an elevated training time complexity. Such large corpora introduce computational challenges at the training stage. Therefore, techniques of bilingual sentence selection (BSS), which choose the most adequate subset of training sentences for a given test set, are appropriate in this case.

A simple selection strategy could simply maximize the coverage of the source part of the test set. But in a translation scenario, a source sentence has potentially many translations. Thereby, to follow this strategy seems inadequate for covering the (unknown) target side of the test set. In order address this issue and include as many eventual translations as possible, the strategy followed by some BSS methods is to increase the informativeness of the selection.

The informativeness of the selection is increased by selecting the most relevant instances of a large dataset. Under an active learning perspective, this large dataset would be the instance pool, from where data is selected. In this scenario, the full training set is considered to be the instance pool. In this work, two BSS strategies which rely on this reasoning are explored: A feature-decay algorithm and an adapted domain adaptation algorithm.

4.1. Feature-Decay Algorithms

Feature-Decay algorithms (FDA) aim to increase the variety of the selection by iteratively choosing sentences whose features are not already included in the selection. Under this scope, a feature is an n-gram.

The process followed by FDA is as follows: First, it is decided a maximum order for the features. The n-grams of the source part of the test set are then extracted. The method considers n-grams from order 1 up to the maximum order. Such features are initially scored, according to an initialization function `init`. Next, the iterative process starts: The sentences belonging to the source part of the training set are scored, according to a sentence scoring function `i`, which depends on the current scores of the features. Note that, because of this, the scores of the sentences will depend only on the n-grams appearing in the test set. The sentence with highest score is then selected. Each time

a sentence is selected, the scores of the features are recomputed, following a `decay` function. This function usually reduces the value of the features already selected. Therefore, it is expected that, in following iterations, different features will be included in the selection. The process continues selecting sentences and decaying features, until the selection has the desired size.

Particular choices in the initialization, scoring and decaying functions provide different selection methods, such as n-gram coverage or TF-IDF (Eck et al., 2005).

4.1.1. FDA5

FDA5 (Biçici and Yuret, 2015) is an instantiation of a feature-decay algorithm, which is controlled under 5 parameters. Features are initialized following:

$$\texttt{init}(\texttt{w}) = \log\left(|\mathcal{X}|/c(\texttt{w})\right)^{i}|\texttt{w}|^{l} \tag{4.1}$$

where $\mathcal{X}$ is the full training corpus and $c(\texttt{w})$ are the counts of the feature $\texttt{w}$ in $\mathcal{X}$. Parameters i and l control the behaviour of the initialization function.

The features decay following a polynomial and an exponential factor, controlled by the parameters c and d. At each iteration, the feature score is recomputed as:

$$\texttt{decay}(\texttt{w}) = \texttt{init}(\texttt{w})(1 + L(\texttt{w}))^{-c}d^{L(\texttt{w})} \tag{4.2}$$

being $L(\texttt{w})$ the number of times that the feature $\texttt{w}$ has been included in the selection.

Finally, score of a sentence $\mathbf{x}$ is the sum of all feature values of the sentence, scaled by a sentence-length factor s:

$$\texttt{i}(\mathbf{x}) = \frac{1}{|\mathbf{x}|^{s}} \sum_{\texttt{w} \in F(\mathbf{x})} score(\texttt{w}) \tag{4.3}$$

where $\texttt{w} \in F(\mathbf{x})$ are the features included in $\mathbf{x}$.

4.2. Infrequent n-gram Recovery

Infrequent n-gram recovery (Gascó et al., 2012) is another BSS technique which follows a similar philosophy than FDA5. The infrequent n-gram recovery technique increases the diversity of the selection by choosing the n-grams with lower appearance frequency in the training corpus.

Similarly to FDA, this method assigns an infrequency score `i` to each sentence, which is updated according the selection process goes on. As in the previous approach, at each iteration, the sentence with the highest score is selected.

In order to control the selection, a infrequency threshold t is set. The selection process runs while at least one sentence $\mathbf{x}$ with infrequency score $\texttt{i}(\mathbf{x})$ lower than t is still in the training set. Note that this stop criterion slightly differs from the one followed by FDA5, where instances were included while the size of the selection is lower than a determined value.

The infrequency score of a source sentence $\mathbf{x}$ is computed as:

$$\mathtt{i}(\mathbf{x}) = \sum_{\mathtt{w} \in F(\mathcal{T})} \min(1, N(\mathtt{w})) \max(0, t - C(\mathtt{w})) \tag{4.4}$$

being $F(\mathcal{T})$ the set of features of the source test set, and $\mathtt{w}$ one of them. $C(\mathtt{w})$ and $N(\mathtt{w})$ are the counts of the feature $\mathtt{w}$ in the whole source training set and in the sentence $\mathbf{x}$, respectively.

This approach requires the rescoring of the full training set at each iteration. Since this can be prohibitively costly, only the sentences with the highest scores are taken into account. This is achieved storing in the list $\mathbf{S}$ only the B sentences with higher score.

The infrequent n-gram recovery method mainly differs from the FDA5 technique in the fact that, in the latter technique, the infrequency score of a sentence depends only on the scores of the features of the sentence, which are extracted from the test set. Hence, as the selection process is carried on, this strategy leads to selections very influenced by the test set. In the case of the infrequent n-gram recovery, the infrequency score is based both on the features of a sentence and on the full training corpus. This makes that the test set has less weight in the selection process, which aims to balance the selection of representative instances from both training and test sets.

Chapter 5

Experiments and Results

In this work, two main experimentation lines were developed. In the first place, the proposed neural language model was tested on a hypotheses rescoring task. For training such model, data had to be selected, according to the aforementioned techniques. Hence, experiments for obtaining the best selection for each task were priorly carried out (Section 5.2). Once the reduced corpus was obtained, the BRNN language model was used to rescore N-best lists of hypotheses (Section 5.3).

The second line referred to neural machine translation. First, experiments on pure NMT were run (Section 5.4). Finally, all translation systems (including the rescored ones) were successfully combined using the confusion network approach (Section 5.5), obtaining the best results of the work.

5.1. Experimentation Framework

Before deepening into each experiment and its results, an brief overview of the main software used in this work is given in this section. In addition, the translation tasks on which the experiments were carried out are also described in Section 5.1.2. For the sake of clarity, all translation results are expressed in %BLEU, although we abuse of notation and refer it just as BLEU.

5.1.1. Software

Two translation toolkits were used for the development of this work. The neural components of the work were built using a Python library. In this section, the main features of such software are briefly highlighted.

Thot

Thot[1] (Ortiz-Martínez and Casacuberta, 2014) is an open-source machine translation toolkit. Thot is still under construction, but it already implements a state-of-art phrase-based translation system and tools to estimate all the statistical models involved in the machine translation process. Moreover, it is able to update incrementally and in real time its models, after receiving a new sample, which makes it appropriate for active and interactive learning scenarios.

[1] `http://daormar.github.io/thot`

Moses

Moses[2] (Koehn et al., 2007) is one of the most popular machine translation toolkits. Nowadays, it is considered as the standard statistical machine translation system. It currently supports phrase-based machine translation, hierarchical phrase-based machine translation and syntax-based translation.

GroundHog

GroundHog[3] is a framework built on the top of Theano[4] (Bastien et al., 2012), a Python library which allows the efficient definition, evaluation and optimization of mathematical expressions involving multi-dimensional arrays. Groundhog eases the building of complex models based on recurrent neural networks. The bidirectional language model developed in this work (Section 2.2), as well as the neural machine translation model (Section 3.1), were built using the tools provided by GroundHog.

5.1.2. Tasks

In this work, the experimentation process was conducted on two tasks: *EU* and *Xerox*. In both tasks, the test set was fixed and known beforehand. All corpora were tokenized and lowercased. The training set was also shuffled, in order to remove eventual correlations in data. For the creation of a development set, sentences were randomly extracted from the training set. The size of the development set was similar to the test set of each respective task. In the following sections, these parallel corpora are described.

Xerox task

The Xerox task consisted in the translation of sentences extracted from the user manuals of Xerox printers. Table 5.1 shows the information related to the partitions. The translation direction was English to Spanish.

		Spanish	**English**
Training	Sentences	56k	
	Running words	747k	662k
	Vocabulary	14k	11k
Development	Sentences	1 012	
	Running words	16k	14k
Test	Sentences	1 125	
	Running words	10k	8k

Table 5.1: Xerox corpus statistics: Number of sentences, words and vocabulary size for each one of the three data sets, training, development and test, for both languages (k and M stand for thousands and millions, respectively).

[2] `http://www.statmt.org/moses`
[3] `http://github.com/lisa-groundhog/GroundHog`
[4] `http://deeplearning.net/software/theano`

EU task

The EU translation task consisted in a selection from the *Bulletin of the European Union* (Khadivi and Goutte, 2003). Table 5.2 shows the information related to the partitions. In this case, the translation was performed from Spanish to English.

		Spanish	**English**
Training	Sentences	214k	
	Running words	5.89M	5.25M
	Vocabulary	54k	40k
Development	Sentences	500	
	Running words	82k	73k
Test	Sentences	800	
	Running words	23k	20k

Table 5.2: EU corpus statistics: Number of sentences, words and vocabulary size for each one of the three data sets, training, development and test, for both languages.

5.2. Results on Data Selection

For selecting an appropriate number of sentences, different configurations for the selection techniques were tested. The test set required by the selection techniques was built appending both development and test sets. For measuring the quality of the selection, n-gram language models were trained over the target side of the selected data and their perplexities were computed. The order of the n-grams ranged from 3 to 5. Since the neural language model only considers a reduced vocabulary, in order to provide a fair comparison in terms of perplexity, the n-gram models were trained using the same vocabulary than the neural model. The selection was chosen looking for a good balance between complexity and quality.

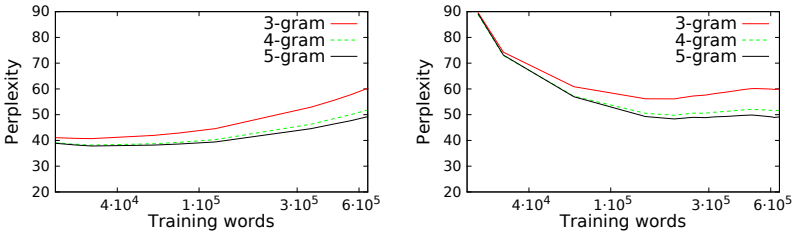
The order of the n-grams used as features in both techniques was set to 5.In the case of FDA5, different configurations of the parameters of the algorithm were tested, sampling the values of the five main parameters (see Section 4.1.1). The best parameter values were found at $i = l = s = 1$, $d = 0.5$, $c = 0$. In the case of the infrequent n-gram recovery technique, the parameter t was sampled in order to provide selections of different sizes.

5.2.1. Xerox

Figure 5.1 shows the perplexity of the different n-gram language models according to the number of selected words in the source part of the corpus. In both selection techniques, 4 and 5-gram models behavioured similarly, while the 3-gram models had a worse performance.

In the case of FDA5, perplexity was slightly increased as more words were selected. This means that FDA5 was able to make good small selections. As the selection size increased, noisy instances were included and hence, the perplexity rose. Infrequent n-gram recovery behaved differently: When required to perform

small selections, the performance was poor. As the sizes of the selections were increased, the method was able to rapidly improve its effectiveness.



(a) FDA5 **(b)** Infrequent n-gram recovery.

Figure 5.1: Perplexity of n-gram language models according to the number of selected words, for both selection techniques.

This is probably because this technique, besides taking into account the source test set for selecting the data, also considers the training corpus. If the infrequency threshold is strict, the technique is forced to capture representative features from both training and test sets with few sentences. That is, the method must maintain a trade-off between representative events from the training corpus and from the test set. Hence, it is likely that the test set had more events which could not be taken into account by small selections. As the infrequency threshold was relaxed, the chances of having a representative selection of the test set were increased and therefore, the performance also rose.

Values of the selection size which offered a good compromise between quality and complexity were found around 120k–150k source words. FDA was hence requested to select 120k source words and the infrequency threshold was set to $t = 10$, which entailed a selection of 147k source words. Table 5.3 shows the details of the selections, compared with the full corpus.

It should be noted that, although infrequent n-gram recovery selected a larger number of words, the number of sentences of the selection was lower than the one provided by FDA5, i.e., the sentences chosen by infrequent n-gram recovery were longer than those selected by FDA5. This phenomenon is produced because infrequent n-gram recovery selects the sentence which most infrequent features have. It is likely that, the longer a sentence is, the more infrequent features has. The criterion followed by FDA5 is based on features scores, being invariant to the sentence length.

		Spanish	**English**
Full training set	Sentences	56k	
	Running words	747k	662k
FDA5 selection	Sentences	9 181	
	Running words	136k	120k
Infrequent n-gram selection	Sentences	8 959	
	Running words	165k	147k

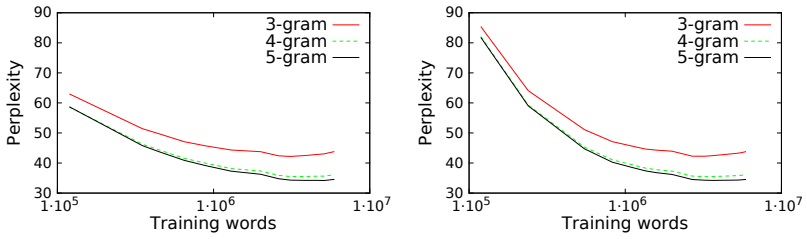
Table 5.3: Statistics for the full EU corpus, for the selection obtained by FDA5 and infrequent n-gram recovery techniques and for the test set.

5.2.2. EU

In Figure 5.2 are shown the perplexities obtained by the different n-gram language models according to the number of source words selected by both selection techniques. Infrequent n-gram recovery behavioured similarly than in the previous task: A bad performance with small selections, which was rapidly enhanced as the infrequency threshold is relaxed. FDA5 depicted a similar conduct, unlike in the previous task.

The differences shown by FDA5 between tasks were due to the difficulty of both tasks: Xerox was an easier one and hence, the selection strategy was able to achieve good results even with small selections. In the EU task, this was not possible: It was necessary to include more events for having a fairly good selection. Even though, FDA5 obtained better selections than infrequent n-gram recovery, which suggests that the first technique is more appropriate for selecting few data.

It is also remarkable that perplexity enhancements were more noticeable when the size of the selection was smaller than one million of words. From here, there were not important differences, which was an expected phenomenon: From one million words ahead, only remained few informative features to add. The inclusion of more instances in the selection did not contribute to add valuable information, therefore, the enhancement in terms of perplexity was low.



(a) FDA5 **(b)** Infrequent n-gram recovery.

Figure 5.2: Perplexity of n-gram language models according to the number of selected words, for both selection techniques.

Considering these results, for selecting the reduced corpus used to train the neural network, FDA5 was asked to select one million of source words. The infrequency parameter of infrequent n-gram recovery was set to $t = 25$. Statistics of the corpora are shown in Table 5.4.

		Spanish	**English**
Full training set	Sentences	213k	
	Running words	5.9M	5.2M
FDA5 selection	Sentences	33k	
	Running words	1M	891k
Infrequent n-gram selection	Sentences	39k	
	Running words	1.3M	1.1M

Table 5.4: Statistics for the full EU corpus, for the selection obtained by FDA5 and infrequent n-gram recovery techniques and for the test set.

5.3. Bidirectional Language Model for Machine Translation

Once the neural language model was trained with the selected data, it was used to rescore N-best lists generated by Thot and Moses translation toolkits.

In the case of Thot, the N-best lists were obtained executing a weight adjustment process, by means of the downhill simplex optimization method (Melder and Nead, 1965), using BLEU as function to maximize. At each iteration of the optimization process, a 200-best list was generated and merged with the list of the previous iteration. The process continued until no new elements were included in the N-best list. The neural language model rescored such lists and another weight adjustment process for the development set was run. At test time, these re-adjusted weights were used.

In the case of Moses, an unique 2500-best list was generated by the decoder. This list was rescored and the weights of the log-linear model were re-estimated running an additional iteration of MERT (Och, 2003). Once again, at test time, the new weights were used.

The hyperparameters of the neural language models (size of the projection layer, size of the recurrent layers and size of the output intermediate layer), were chosen following a perplexity minimization criterion. The learning rate was initially set to 1 and it was halved at the start of each training epoch if the validation entropy did not decrease a 0.3% with respect the previous one (Mikolov, 2012). Following Pascanu et al. (2014), the network was initialized using the standard deviations of a Gaussian white noise distribution.

The neural language model was used solely and also linearly interpolated with the n-gram language model from the original system, following the expression:

$$p_{inter}(y_1^I) = \lambda p_{BRNN}(y_1^I) + (1-\lambda)p_n(y_1^I) \tag{5.1}$$

where $p_{BRNN}(y_1^I)$ and $p_n(y_1^I)$ are the probabilities assigned to the sentence y_1^I by the BRNN and n-gram language models respectively. $p_{inter}(y_1^I)$ is the resulting interpolated probability of the sentence y_1^I. $\lambda \in [0,1]$ is the interpolation coefficient, which determines the importance given to the neural model with respect to the n-gram model. λ was determined by sampling in the development set. The sampling interval was $[0.1, 0.9]$, with a step of 0.1.

5.3.1. Xerox

Table 5.5 reports the perplexities obtained by the different language models. For obtaining a valid comparison of perplexity, all models were trained with the same vocabulary. The shortlist used by the neural model had a size of $\mathcal{K} = 6,000$. The neural network had a projection layer of 500 units, forward and backward layer of 120 units and intermediate output layer of 240 units.

Language model	Perplexity
Full 5-gram	49.1
FDA5 5-gram	39.4
Infrequent 5-gram	49.3
FDA5 BRNN	51.3
Infrequent BRNN	70.1

Table 5.5: Perplexity obtained by the different language models for the Xerox task. Full 5-gram row refers to a 5-gram trained over the complete corpus. The rest of language models are trained over the selection provided by the different techniques.

The bidirectional neural model trained with the data selected by the FDA5 method obtained a similar perplexity than a 5-gram language model trained with all available data. On the other hand, the selection given by infrequent n-gram recovery was useless for training a competitive neural model: An important loss on perplexity was produced.

This was also reflected on the translations results. As shown in Table 5.6, the neural network trained with the data provided by infrequent n-gram recovery, worsened the performance; the interpolation with an n-gram language model could neither achieve enhancements. The reasons for this are probably found in the particularities of the task commented above (Section 5.2.1).

However, the neural model trained with the FDA5 selection did enhance the translation quality, in the case of being interpolated with an n-gram language model. The optimal values for the interpolation weight were found at $\lambda = 0.5$ in the case of the FDA5 network, for both translation toolkits; in the case of infrequent n-gram recovery, the best values were found at $\lambda = 0.2$ in the case of Thot and $\lambda = 0.1$ in the case of Moses. Note that in these latter cases, since the network was unable to add valuable information, the systems worked better when a low importance was given to the neural model, which represented a source of noise.

Language model	Thot	Moses
5-gram	58.6	63.8
FDA5 BRNN	58.2	63.1
Infrequent BRNN	57.2	61.9
FDA5 BRNN + 5-gram	**60.1**	**64.4**
Infrequent BRNN + 5-gram	58.5	63.8

Table 5.6: Test set BLEU score for the different language models and selection techniques. The "+ 5-gram" suffix indicates that the neural language model was linearly interpolated with the original 5-gram language model.

5.3.2. EU

In this task, the vocabulary was limited to 10,000 words. The neural network language model had a size of the projection layer of 620 units, the forward

and backward layers had 200 units each one and the intermediate output layer contained 400 units. Again, for obtaining a valid comparison of perplexity, all models were evaluated over the same vocabulary.

Table 5.7 shows the perplexities obtained by the different language models. The neural models presented perplexities slightly higher than the n-gram language models. Attending to the different selection techniques, the neural language model trained with the FDA5 selection performed better than the one trained with the infrequent n-gram recovery corpus, but such differences were smaller than in the previous task.

Language model	**Perplexity**
Full 5-gram	34.6
FDA5 5-gram	38.7
Infrequent 5-gram	37.4
FDA5 BRNN	53.2
Infrequent BRNN	59.1

Table 5.7: Perplexity obtained by the different language models. Full n-gram row refers to an n-gram trained over the complete corpus. Both neural models were trained over selected instances. In order to provide a fair comparison in terms of perplexity, all models were trained using the same vocabulary (10,000 words).

Regarding the translation quality, Table 5.8 shows the BLEU scores obtained by the different language models for the test set. The optimal values for the interpolation weight were found at $\lambda = 0.6$ in the case of Thot (for both selection techniques), at $\lambda = 0.5$ in the case of Moses with FDA5 and $\lambda = 0.4$ in the case of Moses with infrequent n-gram recovery.

Language model	**Thot**	**Moses**
5-gram	46.1	49.4
FDA5 BRNN	45.2	48.4
Infrequent BRNN	44.7	48.1
FDA5 BRNN + 5-gram	**47.5**	**50.3**
Infrequent BRNN + 5-gram	47.1	50.0

Table 5.8: Test set BLEU score for the different language models and selection techniques.

The results show a similar behaviour of the BRNN language model trained with a FDA5 selection than in the Xerox task: The sole use of the network worsened the performance of the system, but a linear interpolation with an n-gram language model enhanced the translation quality. In this case, the selection provided by infrequent n-gram recovery was closer to the test set. This was reflected either in the perplexity and in the BLEU scores. Although the translation quality was still slightly worse than FDA5, the system was also enhanced with the use of the BRNN language model trained with the infrequent n-gram recovery selection.

5.3.3. Conclusions on the Bidirectional Language Model and Data Selection

In all experiments, the best results were obtained by performing a linear interpolation of the neural language model with an n-gram language model. The sole use of the neural language model worsened the performance of the system. This was probably because neural models have issues with seldom seen events. If a word appears very infrequently, the estimation of the parameters of the network is poor. The interpolation of models aimed to overcome this problem: The flaws of the network with rare words were corrected by the n-gram language model. In addition, since the n-gram model was trained with all available data, it acted as a back-off model, helping the neural model to cope with the reduction of the training corpus. As result, the interpolation of models was able to improve the performance of the system.

Hence, results showed that both approaches, neural and n-gram-based, were complementary: Because of their nature, n-gram language models were robust modellers of local dependencies. The neural network introduced additional information, which was usefully profited by system. Although differences in the results obtained were statistically non-significant, a trend in them was observed. The fact that both systems (Moses and Thot) were enhanced confirms this tendency.

With regard to the selection techniques, it was observed that FDA5 provided better results in all cases than infrequent n-gram recovery. The latter technique suffered a decreasing in the performance when the training and test sets were dissimilar. This was the case of the Xerox task, where none enhancements were obtained. In the case of the EU task, the selection provided was more representative of the test set, and hence, some improvements over the baseline system were found.

Meanwhile, FDA5 found adequate selections for each translation task. Since it considered the test set, it was able to adapt to the requirements of such corpus and it was not influenced by the training set and its noise, as infrequent n-gram recovery did.

This suggests that infrequent n-gram recovery is more sensible to dissimilarities between training and test sets. Hence, it should be used in cases where the training and test sets are meant to be similar or when the test set is unknown beforehand, therefore, the process must be guided by the training set. FDA5, due to the strategy which follows, is more flexible to differences in the training and test sets and provide selections which fit better the test set. Because of this, in our experimental setup, it obtained better results.

5.4. Neural Machine Translation

The same translation tasks were tackled by means of the NMT system. The model had a large set of hyper-parameters which needed to be experimentally tuned. The more sensible hyper-parameters were the following: Size of the vocabulary, size of the word embedding, number of hidden units of the recurrent layers of the encoder and decoder RNN (we used the same number of units for both the encoder and the decoder).

Table 5.9 shows the configurations of the best models for both tasks. In the

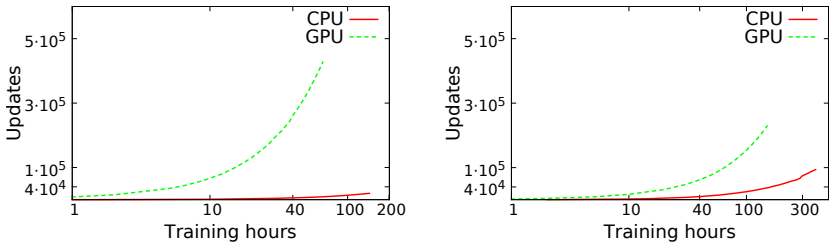
case of the Xerox task, the full vocabulary was taken into account, since it had a tractable size. In the case of the EU task, the short-list approach was used (as done in the case of the neural language model). The words which were not included in the short-list were mapped to the special token `<UNK>`. The size of the short-list was $\mathcal{K} = 11,000$ words, which had a coverage of approximately the 98% of the training corpus.

	Xerox	**EU**
Source vocabulary	11k	11k
Target vocabulary	14k	11k
Source corpus coverage	100%	97.6%
Target corpus coverage	100%	98.6%
Word embedding	520	620
# Hidden units	500	1000
BLEU	55.3	41.6

Table 5.9: Main NMT hyper-parameters and results for both tasks.

On the need of GPUs for training deep models

The training complexity of the NMT model is elevated. Working with deep neural networks requires the usage of graphics processing units (GPU), which greatly boost the computation power. Thanks to the Nvidia Hardware Grant program, we were able to run the NMT experiments on a Nvidia Titan X GPU. This enormously accelerated the training process.



(a) Xerox task. **(b)** EU task.

Figure 5.3: GPU vs CPU performance in the training of the neural model for MT. It was observed a boost of approximately 10× when using the GPU.

In order to illustrate this acceleration, Figure 5.3 shows the differences in speed obtained by using a GPU or a CPU. It is plotted the number of training updates of the model with respect to the time. The models are those from Table 5.9. The experiments on CPU were run using 4 Intel Xeon E5-2450 cores, which made use of the Intel MKL optimization libraries. Using a GPU instead of a CPU greatly boosted the process ($\sim 10\times$).

The increasing of the computation power allowed to a better training of the model, yielding to enhancements up to 14 and 8 BLEU points with respect to the CPU training, on the Xerox and EU tasks respectively.

5.4.1. Conclusions on Neural Machine Translation

Although promising, neural machine translation still underperformed classic phrase-based systems. As main flaws can be highlighted the handling of rare words and the vocabulary limitation. In the first case, NMT systems can only afford the usage of a limited vocabulary. Out-of-vocabulary (OOV) words are mapped to the `<UNK>` token. Therefore, NMT systems are unable to translate OOV words. In the case of the handling of rare words, if a word has been seen seldom times, it will be probably poorly estimated. This can derive in confusion between words, close in the continuous space but far in the meaning. Such confusion leads to erroneous translations.

Phrase-based systems suffer less from these problems, because 1. they can use a much larger vocabulary, having less OOV words and 2. PB systems are able to correctly translate extremely rare events, since they are based on counts and can remember such singularities. Some works already noticed such problems and tried to overcome them, by pre- and post-processing the training data (Luong et al., 2014), sampling a large target vocabulary (Jean et al., 2014) or including an external LM to guide the translation process (Gulcehre et al., 2015). While "vanilla" NMT could not beat the existing PB systems, the above mentioned works were able to obtain better results than the existing state-of-the-art PB system.

NMT has also other minor issues which may difficult its usage. In the first place, the model has a considerable set of hyperparameters which are task-dependent and must be experimentally chosen. In addition, the training complexity of the model makes cumbersome an adequate sampling of the hyperparameters. Because of this, the usage of NMT systems require the use of a GPU, in order to accelerate computations. Otherwise, the process can be prohibitively long.

5.5. System Combination

The combination experiments were performed between three MT systems: Moses, Thot and NMT. In addition, the best rescored versions (from Section 5.3) were also included as additional systems. Hence, the combination module considered 5 systems. For obtaining the best configuration for each task, different orders of the n-gram language model trained over the hypotheses were explored. The sampling interval was from 2 to 5. It was also added an additional n-gram language model, trained over the target training corpus of each task. The order of this extra language model was the same than the one used as base feature.

5.5.1. Xerox

Figure 5.4 shows the results obtained by the different configurations of the combination process. The combination of systems always enhanced the performance of the baseline system. The baseline was determined by the best single system, in this case, Moses rescored with the bidirectional LM.

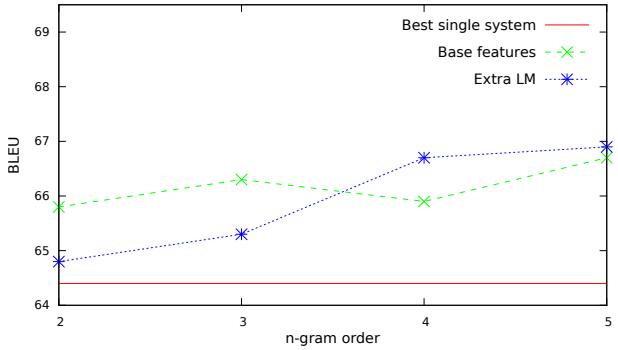


Figure 5.4: Results of the system combination varying the order of the n-gram for the Xerox task. The best single system was obtained by the rescoring of N-best lists from Moses.

The inclusion of an additional language model was beneficial when the order of the n-gram was larger than 3. With lower orders, the additional model only added noise.

This was probably due to particularities of the test set: Since the sentences belonging to this set were quite short, averaging 7 words per sentence, only few long-term dependencies could be found. The existing short-term relationships could be correctly modelled by a low-order n-gram (2 and 3-gram). Hence, low-order n-gram models worked reasonably well. The information added by a low-order extra LM did not helped the system because short-term relationships were already captured by the baseline model. Therefore, the inclusion of a language model trained over the full corpus distorted the original estimations.

When higher-order extra LM were included, the performance of the system was enhanced, since it included longer-term relationships. Since the test set was made up of short sentences, the estimation of longer-term relationships (4 words) was improvable. The additional LM, trained over more data, aided to the estimation of such events, enhancing the performance of the system. Table 5.10 shows the best results obtained by each configuration.

System	**BLEU**
Moses	63.8
Moses + BRNN	64.4
Thot	58.6
Thot + BRNN	60.1
Neural MT	55.3
System combination	66.7
+ Extra language model	**66.9**

Table 5.10: BLEU of the combination of MT systems for the Xerox task.

5.5.2. EU

Figure 5.5 shows the results of the combination of systems, varying the order of the n-gram language model, for both the base and the additional language model. In this case, a smoother behaviour was observed. The additional LM included in every case relevant information and hence, provided enhancements with respect the analogous configuration without extra LM.

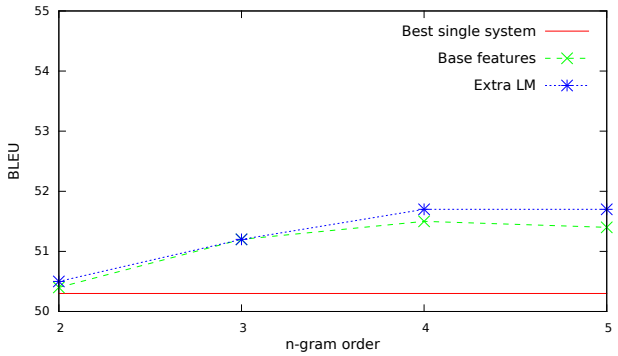


Figure 5.5: Results of the system combination varying the order of the n-gram for the EU task. The best single system was obtained by the rescoring of N-best lists from Moses.

In this task, high-order language models had a better performance than lower-order LM, unlike in the previous task. This is because the sentences in the EU test set are much larger than the test sentences of the Xerox task (25 versus 7 words per sentence, respectively). Hence, higher-order LM could be fully profited in the EU task, while the Xerox task had less chances of having regularities beyond 2 words. Table 5.11 shows the performance of the single systems and the best result achieved by the combination of systems, with and without the extra LM.

System	**BLEU**
Moses	49.4
Moses + BRNN	50.3
Thot	46.1
Thot + BRNN	47.5
Neural MT	41.6
Basic system combination	51.5
+ Extra language model	**51.7**

Table 5.11: BLEU of the combination of MT systems for the EU task.

5.5.3. Conclusions on System Combination

The experiments conducted showed a good behaviour of the combination of MT systems. Since the systems to be combined had a different nature (PB, PB

+ neural rescoring, NMT), the nature of the translations produced was different. This allowed the combination system to add information from diverse sources, which was useful for enhancing the global translation quality. Moreover, the inclusion of an extra language model, trained over larger corpora may help to enhance the systems. The additional LM contributed with information which, although missing in the test set, was valuable for obtaining better translations.

Finally, it was observed that the optimal order of the n-gram language models which produced the best results depended on each test set. In the case of a set with short sentences (like the Xerox task), low-order models worked better than higher-order models, because the estimation of large n-grams could not be correctly performed. On the other hand, if the test set was made up of long sentences (EU task), high-order models overcame lower-order models, since more long-term relationships could be found and high-order n-grams could properly take them into account.

Chapter 6

Future Work and Conclusions

6.1. Future Work

The use of continuous representations of words and phrases is nowadays a hot topic in the natural language processing research community. Perspectives on the application of continuous models and deep learning techniques are encouraging. Nowadays, a big research effort is being spent in the deployment of models for tackling almost every field of the discipline.

In the machine translation area the use of continuous models is also widespread. But it is still required deeper research, in order to exploit the full potential of such powerful models.

Enhancing the NMT

Neural machine translation has some issues (usage of large vocabularies, handling of unknown words, training difficulty), which, although they are being actively tackled, must still be improved. Some works, which address such drawbacks, already beat a state-of-the-art phrase-based system. These results support the neural approach to MT, even though it is still needed more investigation.

Other Structured Prediction Problems

Machine translation belongs to the so-called structured prediction problems. Such techniques aim to predict structured objects (e.g. natural language strings, parse trees, graphs, etc.), rather than scalar or discrete values (as in the case of classification or regression methods). Besides MT, the encoder-decoder approach results appropriate for other generic structured prediction problems. Other tasks are currently being tackled using this approach: The description of multimedia content, like images (Vinyals et al., 2014; Xu et al., 2015) or videos (Venugopalan et al., 2015); conversational modelling (Vinyals and Le, 2015), speech recognition (Chorowski et al., 2015), etc. To explore such research lines would also be an interesting work.

6.2. Conclusions

In this thesis, a language model was developed, based on a bidirectional neural network. It allowed the system to profit from both the past and the future context of words. The training complexity of such model is elevated. For reducing the training time, we explored and compared two different selection techniques, FDA5 and infrequent n-gram recovery. The first one behavioured better than the latter one, because it was focused on a known test set. Infrequent n-gram recovery considered also the training set, producing more noisy selections (for this given test set).

The results obtained in the translation task showed that the neural language model, trained with an appropriate subset of all available data, improved the system, when it was interpolated with an n-gram language model. The inclusion of the n-gram language model was beneficial because of two main reasons: First, since the n-gram language model was trained with all data, it backed off the neural model, addressing the bias introduced by the selection. And second, both models complement naturally: The n-gram language models are robust modellers of short-term relationships, while recurrent neural networks are able to consider larger contexts. Therefore, a combination of the models can profit from these advantages.

Moreover, we made use of the neural approach to machine translation, which relies on the mapping of the source language into a continuous space and its posterior decoding, from this continuous space to the target language space. According to our experiments, NMT performed worse than classic phrase-based approach (in terms of BLEU). Nevertheless, the neural approach could provide additional translation knowledge. For testing this, we conducted experiments on system combination, considering three different approaches to MT: Phrase-based, neural rescoring of PB systems and NMT. Results on the combination experiments showed improvements over the best single system in all experiments. The combination module can be seen as an interpolation between systems, which combines discrete and continuous spaces. As commented above, a combination of different techniques can be positive, because each approach entails its own strengths and weaknesses. The combination aims to maximize the strengths while minimizing the weaknesses.

Another conclusion drawn from this work relates to the computational challenges that such complex systems present. A workaround for overcoming this issue is the development of techniques which allow to work with smaller datasets while trying to keep the performance of a complete corpus. However, for exploiting the power of continuous models to the full, the use of a GPU becomes mandatory.

Finally, some of the work developed in this thesis was submitted and accepted (Peris and Casacuberta, 2015) in the XXXI edition of the congress of the *Sociedad Española para el Procesamiento del Lenguaje Natural* (SEPLN).

Symbols and Acronyms

A. Mathematical Symbols

$\mathcal{F}$	: Source language vocabulary
$\mathcal{E}$	: Target language vocabulary
x_1^J	: Source sentence of length J
y_1^I	: Target sentence of length I
$\hat{y}_1^{\hat{I}}$	: Estimated target sentence
$\mathcal{L}$	: Log-likelihood function
Θ	: Set of parameters of a model
a_j	: Position in y_1^I aligned with position j of x_1^J
$\mathcal{A}(x_1^J, y_1^I)$	: Set of all possible alignments between x_1^J and y_1^I
$c(\cdot)$	: Number of occurrences
$t(\cdot)$	: Alignment model
$l(\cdot)$	: Statistical dictionary
$\widetilde{x}$	: Source phrase
$\widetilde{y}$	: Target phrase
K	: Number of phrases in which a sentence is divided
μ	: Source sentence segmentation
γ	: Target sentence segmentation
α	: Phrase alignment
$h_m(x_1^J, y_1^I)$	: Feature function in a log-linear model
λ_m	: Weight of a feature function in a log-linear model
λ	: Linear interpolation weight
$\mathtt{w}$	: n-gram
ϵ	: Empty string
$\mathbf{h}_t$	: RNN hidden layer at timestep t
$\mathbf{h}_t^{\mathrm{f}}$	: BRNN forward layer at timestep t
$\mathbf{h}_t^{\mathrm{b}}$	: BRNN backward layer at timestep t
$\mathcal{K}$	: Size of the short-list
m	: Dimension of the word embedding
$\mathbf{c}$	: Encoder-decoder context vector
$a(\cdot)$	: Soft alignment model
$\mathcal{B}$	: Beam size
$F(\cdot)$	: Feature extraction function
$\mathcal{X}$	: Source language training corpus
$\mathcal{T}$	: Source language test set

B. List of Acronyms

BLEU	: BiLingual Evaluation Understudy
BRNN	: Bidirectional Recurrent Neural Network
CN	: Confusion Network
FDA	: Feature-Decay Algorithm
GPU	: Graphics Processing Unit
HMM	: Hidden Markov model
LM	: Language Model
MERT	: Minimum Error Rate Training
MT	: Machine Translation
NMT	: Neural Machine Translation
OOV	: Out-Of-Vocabulary
PB	: Phrase-Based
RNN	: Recurrent Neural Network
SMT	: Statistical Machine Translation
TM	: Translation Model

List of Tables

List of Figures

Bibliography

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. Technical report, arXiv preprint arXiv:1409.0473.

Baltescu, P., Blunsom, P., and Hoang, H. (2014). OxLM: A neural language modelling framework for machine translation. *The Prague Bulletin of Mathematical Linguistics*, 102(1):81–92.

Bangalore, S., Bordel, G., and Riccardi, G. (2001). Computing consensus translation from multiple machine translation systems. In *Automatic Speech Recognition and Understanding, 2001. ASRU'01. IEEE Workshop on*, pages 351–354. IEEE.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Machine Learning Research*.

Biçici, E. and Yuret, D. (2011). Instance selection for machine translation using feature decay algorithms. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 272–283. Association for Computational Linguistics.

Biçici, E. and Yuret, D. (2015). Optimizing instance selection for statistical machine translation with feature decay algorithms. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(2):339–350.

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.

Castaño, M. A., Casacuberta, F., and Vidal, E. (1997). Machine translation using neural networks and finite-state models. *Theoretical and Methodological Issues in Machine Translation (TMI)*, pages 160–167.

Chen, S. F. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard U., Cambridge, MA.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. *CoRR*, abs/1506.07503.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

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. Association for Computational Linguistics.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

Eck, M., Vogel, S., and Waibel, A. (2005). Low cost portability for statistical machine translation based on n-gram frequency and TF-IDF. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 61–67.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Fiscus, J. G. (1997). A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 347–354. IEEE.

Freitag, M., Huck, M., and Ney, H. (2014). Jane: Open source machine translation system combination. In *Proc. of the Conf. of the European Chapter of the Assoc. for Computational Linguistics (EACL), Gothenburg, Sweden*, pages 29–32.

Gascó, G., Rocha, M.-A., Sanchis-Trilles, G., Andrés-Ferrer, J., and Casacuberta, F. (2012). Does more data always yield better translations? In *Proceedings of the 13th European Chapter of the Association for Computational Linguistics*, pages 152–161.

González-Rubio, J., Juan, A., and Casacuberta, F. (2011). Minimum bayes-risk system combination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1268–1277. Association for Computational Linguistics.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv:*`1308.0850 [cs.NE]`.

Gulcehre, C., Firat, O., Xu, K., Cho, K., Barrault, L., Lin, H.-C., Bougares, F., Schwenk, H., and Bengio, Y. (2015). On using monolingual corpora in neural machine translation. *arXiv:1503.03535*.

Hasler, E., Haddow, B., and Koehn, P. (2011). Margin infused relaxed algorithm for moses. *The Prague Bulletin of Mathematical Linguistics*, 96:69–78.

Heafield, K., Hanneman, G., and Lavie, A. (2009). Machine translation system combination with flexible word ordering. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 56–60. Association for Computational Linguistics.

Hopkins, M. and May, J. (2011). Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362. Association for Computational Linguistics.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2014). On using very large target vocabulary for neural machine translation.

Jordan, M. I. (1990). Artificial neural networks. chapter Attractor Dynamics and Parallelism in a Connectionist Sequential Machine, pages 112–127. IEEE Press, Piscataway, NJ, USA.

Khadivi, S. and Goutte, C. (2003). Tools for corpus alignment and evaluation of the alignments (deliverable d4.9). Technical report, Technical report, TransType2 (IST-2001-32091).

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., 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.

Koerner, E. F. K. and Asher, R. E. (2014). *Concise history of the language sciences: from the Sumerians to the cognitivists*. Elsevier.

Luong, T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation.

Matusov, E., Leusch, G., Banchs, R. E., Bertoldi, N., Dechelotte, D., Federico, M., Kolss, M., Lee, Y.-S., Marino, J. B., Paulik, M., et al. (2008). System combination for machine translation of spoken and written language. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(7):1222–1237.

Melder, J. A. and Nead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4):308–313.

Mikolov, T. (2012). *Statistical Language Models based on Neural Networks*. PhD thesis, Brno University of Technology.

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.

Ortiz-Martínez, D. (2011). *Advances in Fully-Automatic and Interactive Phrase-Based Statistical Machine Translation*. PhD thesis, Universidad Politécnica de Valencia.

Ortiz-Martínez, D. and Casacuberta, F. (2014). The new Thot toolkit for fully-automatic and interactive statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 45–48.

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

Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. (2014). How to construct deep recurrent neural networks.

Peris, Á. and Casacuberta, F. (2015). A bidirectional recurrent neural language model for machine translation. *Procesamiento del Lenguaje Natural*, 55:109–116.

Rosti, A.-V., He, X., Karakos, D., Leusch, G., Cao, Y., Freitag, M., Matsoukas, S., Ney, H., Smith, J., and Zhang, B. (2012). Review of hypothesis alignment algorithms for mt system combination via confusion network decoding. In *NAACL 2012 Seventh Workshop on Statistical Machine Translation*, pages 191–199, Montreal, Canada.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Schwenk, H. (2013). CSLM - a modular open-source continuous space language modeling toolkit. In *INTERSPEECH*, pages 1198–1202. ISCA.

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.

Stolcke, A. (2002). Srilm - an extensible language modeling toolkit. pages 901–904.

Sundermeyer, M., Alkhouli, T., Wuebker, J., and Ney, H. (2014). Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 14–25. Association for Computational Linguistics.

Sundermeyer, M., Schlüter, R., and Ney, H. (2012). LSTM neural networks for language modeling. In *Interspeech*, pages 194–197.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Venugopalan, S., Xu, H., Donahue, J., Rohrbach, M., Mooney, R., and Saenko, K. (2015). Translating videos to natural language using deep recurrent neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1494–1504. Association for Computational Linguistics.

Vinyals, O. and Le, Q. V. (2015). A neural conversational model. *CoRR*, abs/1506.05869.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2014). Show and tell: A neural image caption generator.

Wang, R., Zhao, H., Lu, B.-L., Utiyama, M., and Sumita, E. (2014). Neural network based bilingual language model growing for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 189–195. Association for Computational Linguistics.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044.

Zens, R., Och, F. J., and Ney, H. (2002). Phrase-based statistical machine translation. In *KI 2002: Advances in Artificial Intelligence*, pages 18–32. Springer.