eman ta zabal zazu

Universidad del País Vasco   Euskal Herriko Unibertsitatea

Distributional Semantics and Machine Learning for Statistical Machine Translation

Author: Mikel Artetxe

Advisors: Eneko Agirre and Gorka Labaka

HAP/LAP

Hizkuntzaren Azterketa eta Prozesamendua
Language Analysis and Processing

Final Thesis

May 2016

Departments: Computer Systems and Languages, Computational Architectures and Technologies, Computational Science and Artificial Intelligence, Basque Language and Communication, Communications Engineering.

Laburpena

Lan honetan semantika distribuzionalaren eta ikasketa automatikoaren erabilera aztertzen dugu itzulpen automatiko estatistikoa hobetzeko. Bide horretan, erregresio logistikoan oinarritutako ikasketa automatikoko eredu bat proposatzen dugu hitz-segiden itzulpen-probabilitatea modu dinamikoan modelatzeko. Proposatutako eredua itzulpen automatiko estatistikoko ohiko itzulpen-probabilitateen orokortze bat dela frogatzen dugu, eta testuinguruko nahiz semantika distribuzionaleko informazioa barneratzeko baliatu ezaugarri lexiko, hitz-cluster eta hitzen errepresentazio bektorialen bidez. Horretaz gain, semantika distribuzionaleko ezagutza itzulpen automatiko estatistikoan txertatzeko beste hurbilpen bat lantzen dugu: hitzen errepresentazio bektorial elebidunak erabiltzea hitz-segiden itzulpenen antzekotasuna modelatzeko. Gure esperimentuek proposatutako ereduen baliagarritasuna erakusten dute, emaitza itxaropentsuak eskuratuz oinarrizko sistema sendo baten gainean. Era berean, gure lanak ekarpen garrantzitsuak egiten ditu errepresentazio bektorialen mapaketa elebidunei eta hitzen errepresentazio bektorialetan oinarritutako hitz-segiden antzekotasun neurriei dagokienean, itzulpen automatikoaz haratago balio propio bat dutenak semantika distribuzionalaren arloan.

Abstract

In this work, we explore the use of distributional semantics and machine learning to improve statistical machine translation. For that purpose, we propose the use of a logistic regression based machine learning model for dynamic phrase translation probability modeling. We prove that the proposed model can be seen as a generalization of the standard translation probabilities used in statistical machine translation, and use it to incorporate context and distributional semantic information through lexical, word cluster and word embedding features. Apart from that, we explore the use of word embeddings for phrase translation probability scoring as an alternative approach to incorporate distributional semantic knowledge into statistical machine translation. Our experiments show the effectiveness of the proposed models, achieving promising results over a strong baseline. At the same time, our work makes important contributions in relation to bilingual word embedding mappings and word embedding based phrase similarity measures, which go beyond machine translation and have an intrinsic value in the field of distributional semantics.

Contents

List of Figures

List of Tables

1 Introduction

Machine translation has been one of the most prominent applications of natural language processing and artificial intelligence since their early days, meeting the dream to break the language barrier in an increasingly global yet diverse world. The classical approach, based on rules, has been progressively replaced by statistical machine translation, which has become the dominant paradigm bringing about a breakthrough to the field. Nevertheless, in spite of the great progress in recent years, current machine translation engines still suffer from important limitations, which are mostly related to the following factors:

- Sparsity of natural language, causing many terms or linguistic phenomena to be missing or poorly represented in the training corpora with the inherent difficulty to properly model them, a problem that is accentuated by the scarcity of bilingual resources for the vast majority of language pairs.
- Locality of the statistical methods used, with the inherent difficulty to properly model context information, long distance dependencies and morphosyntactically diverging language pairs.
- Inflexibility of the statistical models used, with the inherent difficulty to incorporate external linguistic information from other natural language processing tools and resources.

In this work, we explore the use of distributional semantic and machine learning techniques to address these issues with the aim to improve machine translation. More concretely, we develop a logistic regression based machine learning model for dynamic phrase translation probability scoring, which we prove to be a generalization of the standard translation probabilities used in statistical machine translation that allows to naturally incorporate additional features to obtain better probability estimates. In this regard, we explore the use of lexical features shared across different phrase pairs that could have a smoothing effect and help to overcome the sparsity problem, source language context features that could help to overcome the locality problem, and additional features for external linguistic information that could help to overcome the inflexibility problem. Taking advantage of this last point, we also incorporate distributional semantic features into the model through word clusters and word embeddings. Based on the distributional hypothesis, which states that the meaning of each word in a language is defined by its usage and can therefore be characterized by the words it is generally surrounded by, distributional semantics attempts to build abstract representations of words based on their usage in a large corpus. In particular, word clustering uses discrete classes to group related words together, whereas word embeddings are dense, low-dimensional representations of words in a continuous vector space. Therefore, these distributional semantic features do not only serve to incorporate unsupervised knowledge acquired from large monolingual corpora into the model, but are also useful to mitigate the sparsity problem. In addition to that, we also explore an alternative method to incorporate distributional semantic information into machine translation through the use of bilingual word embeddings for phrase translation similarity scoring. Some of our contributions in this direction go beyond machine translation and have an intrinsic value in distributional semantics. In particular, we develop a general framework to learn bilingual word embedding mappings and show that several existing methods fit naturally in it while revealing flaws in their theoretical justification. At the same time, we analyze the centroid cosine similarity from a theoretical perspective and propose alternative word embedding based phrase similarity measures that address some of its issues.

This dissertation is structured as follows. From Chapter 2 to 4, we describe the background that serves as the basis for our work in statistical machine translation, large scale machine learning and distributional semantics, respectively. More concretely, Chapter 2 presents the different machine translation paradigms, describes statistical machine translation in depth, outlines machine translation evaluation, and introduces the related software used in the project. Chapter 3 then discusses supervised machine learning, describes linear models with a focus on logistic regression and gradient descent, explains the limitations of these models and different approaches to address them, discusses the peculiarities of large scale machine learning, and presents the Vowpal Wabbit large scale machine learning system used in the project. After that, Chapter 4 discusses distributional semantics, describing monolingual word embeddings, word clustering and bilingual word embeddings as well as existing methods to integrate them in machine translation.

Chapter 5 through 8 then focus on our own contributions to the field. More concretely, Chapter 5 describes the logistic regression model we propose for phrase translation probability scoring, proves its equivalence with the relative frequency counting probability estimates used in statistical machine translation, presents a set of lexical features for it, and experimentally tests them on English-Spanish machine translation. Chapter 6 then discusses the incorporation of distributional semantic features into the model through word clusters and word embeddings and tests them on the same English-Spanish machine translation task. After that, we present our general framework for bilingual word embedding mappings in Chapter 7, which starts with a basic optimization objective and allows for several variants that we prove to be equivalent to other meaningful optimization goals, we analyze its relation to other related methods proposed in the literature, and test them in English-Italian word translation induction. Chapter 8 then discusses the use of bilingual word embeddings for phrase translation similarity scoring, analyzing the standard centroid cosine similarity from a theoretical perspective, proposing alternative phrase similarity measures that address some of its issues and experimentally testing them on a new phrase translation selection task we propose as well as on English-Spanish end-to-end machine translation. Finally, Chapter 9 outlines the conclusions drawn from our work and suggests futures lines of research.

2 Statistical machine translation

The goal of machine translation (MT) is to automatically translate from one natural language to another using computers. It has been one of the most prominent applications of natural language processing since its early days as, in addition to being a very complete problem, it has a great practical interest in an increasingly global world.

Different approaches have been followed to achieve this goal, which have been broadly classified into two main groups: **rule-based machine translation** systems, which follow a rational approach taking our linguistic knowledge of the languages involved as their basis, and **corpus-based machine translation** systems, which follow an empirical approach taking previously made translations as their basis. At the same time, the latter are divided into **example-based machine translation** systems that translate by analogy, and **statistical machine translation** systems that make use of different statistical models for the translation process. The statistical approach has progressively replaced previous attempts since the 1990s until becoming the main approach to MT nowadays (Hutchins, 2007), and it is also the approach that has been followed in this work.

The remaining of this chapter is organized as follows. Section 2.1 presents a more detailed discussion of the different MT paradigms. Section 2.2 explains how word alignment is done, which is the basis of most statistical machine translation systems. Section 2.3 then introduces the phrase-based statistical machine translation paradigm, which has been the main approach to statistical machine translation and also the one used in this work. Section 2.4 discusses the main limitations of phrase-based systems and presents alternative approaches that have arisen addressing them. After that, Section 2.5 outlines MT evaluation. Section 2.6 then presents Moses, the most widely extended statistical machine translation toolkit and also the one we use in this project. Finally, Section 2.7 concludes the chapter.

In addition to the specific citations in the text, Jurafsky and Martin (2008) and, to a lesser extent, Labaka (2010) and Manning and Schütze (1999) have been used as a general reference for writing this chapter.

2.1 Machine translation paradigms

As introduced previously, there are two main machine translation paradigms: rule-based machine translation and corpus-based machine translation, which is further divided into example-based machine translation and statistical machine translation. This section gives a brief overview of each of these paradigms, starting with rule-based approaches in Section 2.1.1 and continuing with corpus-based ones in Section 2.1.2. Finally, Section 2.1.3 discusses hybrid approaches that try to combine different paradigms.

2.1.1 Rule-based machine translation

Rule-based machine translation (RBMT) makes use of the linguistic knowledge of the source and the target languages as the basis to tackle machine translation. Depending on

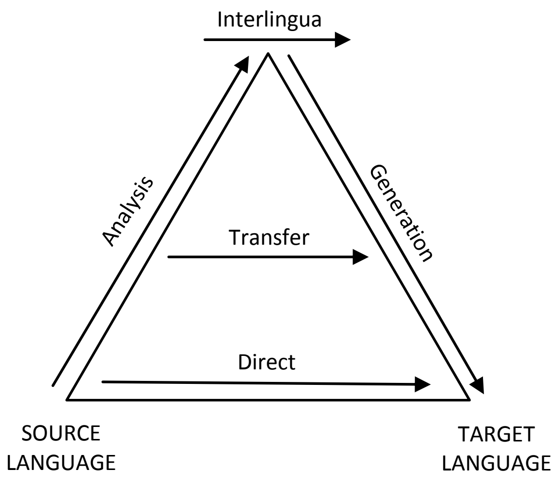

Figure 1: The Vauquois triangle

the nature of this knowledge, the abstraction level, and the intermediate representation used, different approaches have been followed, which have been classified into direct systems, transfer-based systems and interlingual systems. The so called Vauquois triangle, shown in Figure 1, has been typically used to illustrate these three strategies. As it can be seen, the more close we move to the top of the triangle, the more complex the intermediate representation used becomes, requiring of a deeper processing. More concretely, each approach works as follows:

- **Direct systems** translate word by word in a single step, without using any intermediate representation. They mainly rely on bilingual dictionaries for that, but basic coherence and reordering rules have also been used in addition to them.

- **Transfer-based system** are based on contrastive knowledge, that is, the knowledge of the differences between the source and the target language, and make use of different rules to overcome them. For that purpose, two intermediate representations are used, one for the source language and the other one for the target language, dividing the translation process into the following three steps:

  – In the **analysis** phase, the text to be translated is analyzed making use of the standard natural language processing pipeline, usually involving morphological analysis, morphological disambiguation or part-of-speech tagging, and syntactic analysis. Depending on the nature of this analysis, these systems are further divided into **shallow-transfer** systems, if only shallow parsing or chunking is done, or **deep-transfer** systems, if the full parse tree is built.

  – During the **transfer** phase, the source language representation is transformed into the target language representation. **Syntactic-transfer** systems do this at the lexical and structural level by means of bilingual dictionaries and transfer rules, whereas **semantic-transfer** systems also work at the semantic level making use of additional structures to represent meaning.

  – In the **generation** phase, the final translation is obtained from the target language representation. Most of the times, morphological dictionaries serve for this purpose.

- **Interlingual systems** make use of a single intermediate representation called interlingua that is independent from any specific language. Thanks to this, the translation process is reduced to two steps: the analysis phase encodes the meaning of the input text into the interlingua, while the generation phase decodes the interlingua into the output text. This way, unlike transfer-based systems, which work at a language pair level, interlingual systems follow a global approach, working at a conceptual level. The main advantage of this is that it makes it much easier to translate among several languages, as it is enough to build the analysis and generation module for each of them in a completely independent way from each other. However, getting a useful interlingua representation has proved to be extremely challenging in practice, which is the reason why the approach has not attracted too much attention in the last decades.

2.1.2 Corpus-based machine translation

The corpus-based machine translation paradigm makes use of previously made human translations as the basis to make new translations. The popularization of new technology and, in particular, the Internet, has made huge amount of text data easily available to the research community which, along with the computational power provided by modern hardware, has motivated the raise of these approaches in the last decades. Parallel corpora (i.e. bilingual texts aligned at segment -usually sentence- level) are the main resource used for that. In particular, the research community has compiled several parallel corpora coming from official documents translated into different languages and published through the web by different public institutions.

As said before, there are two main paradigms that follow this approach, the so called example-based machine translation and statistical machine translation, which are further discussed in the following two subsections.

2.1.2.1 Example-based machine translation

Example-based machine translation (EBMT) systems translate by analogy, using a parallel corpus at runtime as the only source of knowledge (Nagao, 1984; Somers, 2003). Many different approaches have been proposed around this idea, generally involving the following three steps:

- **Matching**, that is, searching for occurrences of the input text in the parallel corpus.

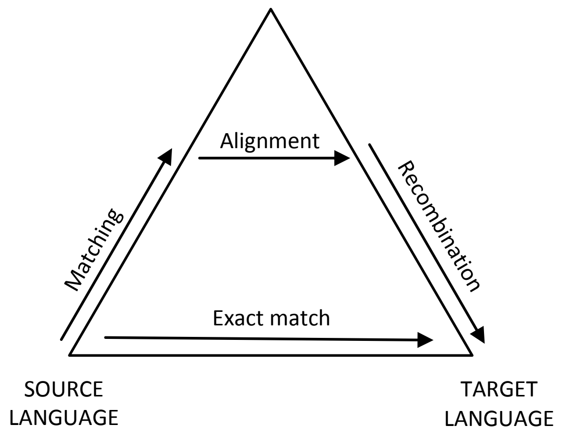

Figure 2: The Vauquois triangle adapted for EBMT systems

- **Alignment**, that is, identifying the translation of each matched chunk in the parallel corpus.
- **Recombination**, that is, combining the aligned chunks to obtain the best possible translation of the input text.

Even though the underlying principle is completely different, these steps resemble the analysis, transfer and generation steps of transfer-based RBMT systems, as they respectively take care of processing the input text, moving it into the target language, and adapting it to obtain the final translation. This way, the Vauquois triangle has been adapted for EBMT systems as shown in Figure 2. As it can be seen, in addition to transfer-based RBMT, EBMT is also analogous to direct RBMT in one trivial case as, when an exact match of the input text is found, its translation in the parallel corpus would be directly used with no further processing.

2.1.2.2 Statistical machine translation

Statistical machine translation (SMT) systems use statistical models learned from parallel corpora to tackle the problem of machine translation. This way, instead of focusing on the translation process itself, this approach starts by looking at the output it should generate first and, given a source language text F, tries to models the probability $P(E|F)$ of any target language text E being its translation. Therefore, the goal of the system will simply be to choose the translation $\hat{E}$ that maximizes this probability:

$$\hat{E} = \arg\max_{E} P(E|F)$$

The initial SMT models worked at the word level, but they were later superseded by phrase-based systems capable of working with larger chunks of text, becoming the most widely used approach to SMT. However, these word level models are still used today to align parallel corpora as explained later in Section 2.2, a necessary step for building phrased-based systems, which are then discussed in Section 2.3.

Several extensions of the phrase-based approach have also been proposed, tree-based models, capable of working with hierarchical structures, and factored models, which provide a natural way to incorporate linguistic features into the SMT system, being the most prominent ones. With the irruption of deep learning, there have also been several attempts in the last few years to replace different components of traditional SMT systems with neural networks, and neural machine translation, which tries to use neural networks to model the entire translation process, has also emerged generating big expectations. All these other approaches are later discussed in Section 2.4.

2.1.3 Hybrid machine translation

Even though SMT has become the main approach to MT, the fact is that each paradigm has its own strengths and weaknesses. For instance, SMT systems tend to produce more natural translations than RBMT systems, but their errors also tend to be more unpredictable from the human perspective. For that reason, there have been several approaches to combine different paradigms through hybridization, which can be broadly classified into the following three groups (Lu and Xue, 2010):

- **Hybrid combination** takes one main system and uses another one to create or improve resources for it. Following this approach, there have been attempts to adapt RBMT rules for SMT systems, and also to improve RBMT systems from parallel corpora using SMT techniques.
- **Multi-engine or parallel combination** translates the input text using several independent systems, and uses an additional module to combine their output in the best possible way.
- **Multi-pass or serial combination** uses the output of one system as the input of another. Automatic post-edition is the most prominent application of this approach, where one system tries to improve the output of another.

2.2 Word alignment

The goal of word alignment is to identify translation relationships among words in a parallel corpus, usually at sentence level. This way, its output is a bipartite graph that links each word in one language with those words in the target language that are considered to be its translation.

In addition to being the foundation of most SMT systems, word alignment has also been used in other crosslingual natural language processing tasks. However, most applications require to align thousands and thousands of sentences for it to be useful, making manual annotation unfeasible. For that reason, unsupervised learning is used instead, making use of different statistical models, IBM Models 1-5 (five models of increasing complexity) (Brown et al., 1993) and Hidden Markov Models (HMM) (Vogel et al., 1996) being the most prominent ones.

All these models are based on one important simplification, as they only allow to align each word in the target language F with a single word in the source language E. In order to make it possible to align words in the target language that do not have a direct correspondence in the source language (known as spurious words), the source sentence is assumed to have the *null* pseudo-token at position 0. This way, an alignment A can be represented as a sequence of integers $A = a_1, a_2, \ldots, a_J$, where a_j denotes the position of the source language word that has been aligned with the jth word in the target language, I and J being the length of the source and the target languages and subject to $0 \leq a_j \leq I$.

Within this framework, given a word sequence E in the source language, each model estimates the probability of obtaining the word sequence F in the target language through any alignment A in one way or another, taking the optimal alignment $\hat{A}$ that maximizes this probability:

$$\hat{A} = \arg\max_{A} P(F, A|E)$$

For instance, **IBM Model 1**, the simplest one among all of them, models the probability of the source language word sequence $E = e_1, e_2, \ldots, e_I$ and alignment $A = a_1, a_2, \ldots, a_J$ generating the target language word sequence $F = f_1, f_2, \ldots, f_J$ as follows, where $t(f_x|e_y)$ denotes the probability of e_y being translated as f_x:

$$P(F|E, A) = \prod_{j=1}^{J} t(f_j|e_{a_j})$$

IBM Model 1 assumes that all the alignments are equally likely. Therefore, since there are $(I+1)^J$ possible alignments for a given target language length J and assuming that the probability of the actual length being J is a small constant ϵ, the probability of the source language word sequence E being aligned as A is estimated as follows:

$$P(A|E) = \frac{\epsilon}{(I+1)^J}$$

And, combining all the equations above, the model would choose the optimal alignment $\hat{A}$ for every entry in the parallel corpus as follows:

$$\begin{aligned} \hat{A} = \arg\max_{A} P(F, A|E) &= \arg\max_{A} P(F|E, A) \times P(A|E) \\ &= \arg\max_{A} \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} t(f_j|e_{a_j}) \end{aligned}$$

Finally, since the alignment of each word is decided independently from the rest, the above equation can be simplified as follows:

$$\hat{A} = \arg\max_{a_j} t(f_j|e_{a_j}) \quad 1 \leq j \leq J$$

The rest of the models try to overcome all those simplifications made by IBM Model 1 and, as a consequence, they turn out to be considerably more complex:

- **IBM Model 2** adds an absolute reordering model, addressing the important limitation in IBM Model 1 of not taking the order of the words into account.
- **IBM Model 3** introduces the notion of fertility, modeling the number of target language words that each source language word generates.
- **IBM Model 4** incorporates a relative reordering model.
- **IBM Model 5** fixes the deficiency problem in IBM Model 3 and 4, which assigned a probability mass to impossible events as a consequence of not preventing words being placed in positions that were already taken.
- **Hidden Markov Models (HMM)** are used as an alternative approach by using the hidden states to represent source words, observations to represent target words and transition probabilities to model alignment probabilities.

In any case, all the models combine the following two components in one way or another:

- The **word alignment** itself, represented as the sequence of integer $A = a_1, a_2, \ldots, a_J$.
- **Lexical weights** or translation probabilities and other parameters of the model, which corresponded to $t(f_x|e_y)$ in the case of IBM Model 1.

Needless to say, if the model parameters were known beforehand, it would be trivial to find the optimal alignment according to them. At the same time, if the alignments were known beforehand, it would be straightforward to estimate the parameters of a model (e.g. taking relative frequency estimates for the translation probabilities). It seems a paradox that computing alignments requires of the model parameters while computing the model parameters requires of alignments. However, if we had some estimation for the model parameters, it would be possible to estimate the alignment probabilities, and then use this estimation to re-estimate the model parameters. This is precisely the idea behind the expectation maximization (EM) algorithm, which initializes all the probabilities uniformly and finds better and better estimates for the alignment probabilities and the model parameters by fixing one to re-estimate the other and the other way around in an iterative process. As this is a never-ending story, only a fixed number of iterations are usually performed. At the same time, the complexity of the most advanced models makes it computationally unfeasible to make the exact calculations for each EM iteration, so the simplest models, which do not have this limitation, are usually used to provide an initial estimate.

Finally, recall that all these models only allow to align a target language word with a single word in the source language. Even though this makes the alignment problem easier to tackle, it might also pose an important limitation on the usefulness of the results, as it

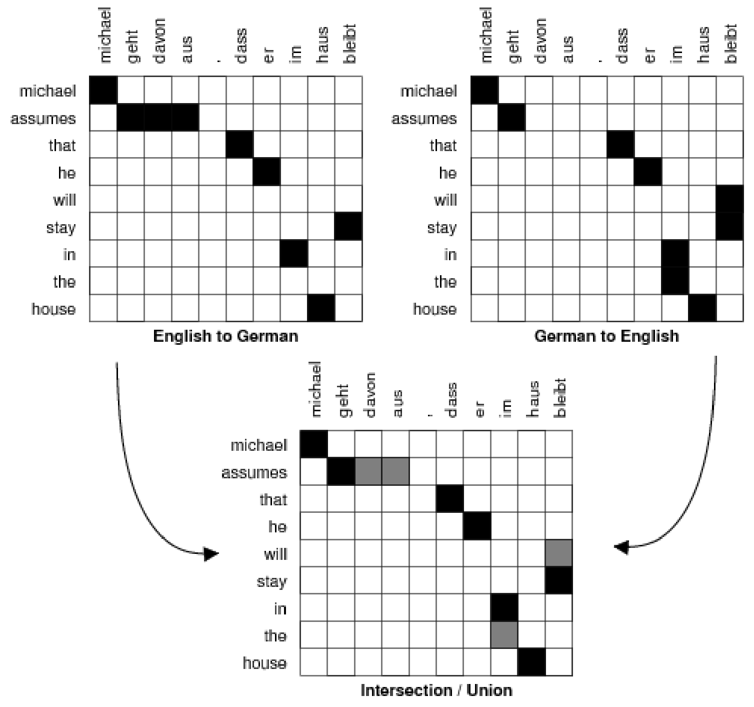

Figure 3: Intersection and union for alignment symmetrization (Koehn, 2016)

is possible that a target language word is actually the translation of several words in the source language. In order to overcome this issue, alignment symmetrization techniques are used, which compute word alignments in each direction independently and then combine their results. Two straightforward options would be to take the intersection or the union between both alignments. The former would likely discard some correct alignments, but those that it keeps would presumably have a very high accuracy. At the same time, the latter would be likely to contain most correct alignments, but the accuracy of the alignments it keeps would presumably be lower. Figure 3 shows the effect of each of them for one example case. Since both options have their own problems, more sophisticated alignment methods have been proposed. These methods usually start from the intersection between both alignments and incorporate additional alignment points from their union according to different heuristics (Och and Ney, 2003).

2.3 Phrase-based statistical machine translation

Following the discussion in Section 2.1.2.2 on SMT systems, phrase-based models (Koehn et al., 2003) try to find the translation $\hat{E}$ that maximizes the probability $P(E|F)$ of being the translation of the input text F. Applying the Bayes' rule, this can be formulated as follows:

$$\hat{E} = \arg\max_E P(E|F) = \arg\max_E \frac{P(F|E)P(E)}{P(F)} = \arg\max_E P(F|E)P(E)$$

As it can be seen, according to the last formulation the selection of the optimal translation $\hat{E}$ depends on two factors: the probability $P(F|E)$ of the input text F being the translation of the output text E, and the probability $P(E)$ of the output text itself occurring in the target language. Original phrase-based models follow this noisy channel approach and use two independent components to model each of them, the so called translation and language models:

$$\hat{E} = \arg\max_{E} \underbrace{P(F|E)}_{\substack{\text{translation}\\ \text{model}}} \underbrace{P(E)}_{\substack{\text{language}\\ \text{model}}}$$

Both the translation and the language models are statistical models in a phrase-based system, whose parameters are estimated from large corpora. Besides them, these systems have an additional component, the decoder, which takes care of searching for the translation that maximizes the aforementioned probability. This is how they work one by one:

- The **translation model** assigns a probability to a given target language sentence being translated as some source language sentence. For that purpose, phrase-based models segment the source and the target sentences into pairs of aligned phrases (sequences of words without any linguistic motivation) and typically calculate the probability in question as follows:

  $$P(F|E) = \prod_{i=1}^{I} \phi(\bar{f}_i, \bar{e}_i) d(a_i - b_{i-1})$$

  where $\phi(\bar{f}_i, \bar{e}_i)$ corresponds to the translation probability and $d(a_i - b_{i-1})$ corresponds to the distortion probability. The former models the probability of the phrase $\bar{e}_i$ in the target language generating the phrase $\bar{f}_i$ in the source language, whereas the latter models the distance between the phrases in both languages. More concretely, a_i denotes the starting position of the source language phrase generated by the ith phrase in the target language, while b_{i-1} denotes the ending position of the source language phrase generated by the $(i-1)$th phrase in the target language. There can be different ways to calculate the distance itself, a simple approach being to raise a small constant α to the distortion:

  $$d(a_i - b_{i-1}) = \alpha^{|a_i - b_{i-1} - 1|}$$

  In order to estimate the translation probabilities $\phi(\bar{f}_i, \bar{e}_i)$, word alignment is first performed in both directions in a parallel corpus and symmetrization is applied as described in Section 2.2. Having done that, all the phrase pairs that are consistent with the resulting alignment points are extracted as shown in Figure 4. Finally, relative frequency estimates are used for the translation probabilities:

  $$\phi(\bar{f}, \bar{e}) = \frac{\text{count}(\bar{f}, \bar{e})}{\sum_{\bar{f}} \text{count}(\bar{f}, \bar{e})}$$

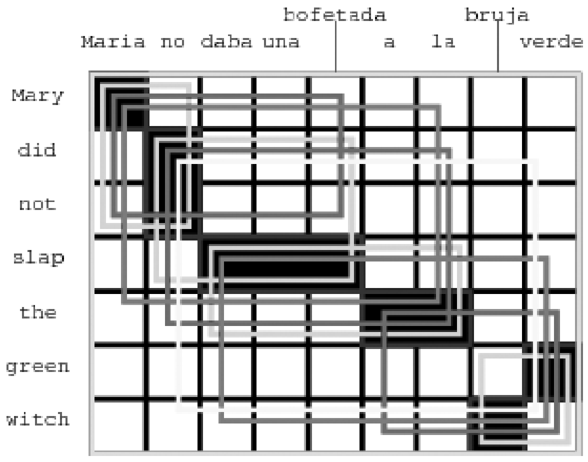

Figure 4: Phrase pair extraction from word alignment (Koehn, 2016)

Typically, all the extracted phrase pairs are stored together with their translation probability in a big data structure known as the phrase table.

- The **language model** assigns a probability to a given sequence of words $w_1, \ldots, w_m$ occurring in the target language. Different models have been proposed for that, but most of them are based on the concept of n-grams. Within this approach, the probability of a sequence of words occurring in the language in question corresponds to the product of the probability of each word occurring after the previous n words, where n is a parameter of the model:

$$P(w_1, \ldots, w_m) \approx \prod_{i=1}^{m} P(w_i | w_{i-(n-1)}, \ldots, w_{i-1})$$

This way, for different values of the parameter n different models like unigrams ($n = 1$, i.e. the probability of each word is independent from the context), bigrams ($n = 2$) and trigrams ($n = 3$) are defined. In order to calculate the conditional probabilities for a given value of n, relative frequency estimates are usually taken from a large monolingual corpus:

$$P(w_i | w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1})}$$

Nevertheless, this model would assign a zero probability to words or n-grams not occurring in the parallel corpus and, in order to overcome this issue, different smoothing and backoff techniques are typically used.

- The **decoder** takes a text in the source language and generates the most probable translation according to the translation and language models. From the algorithmic

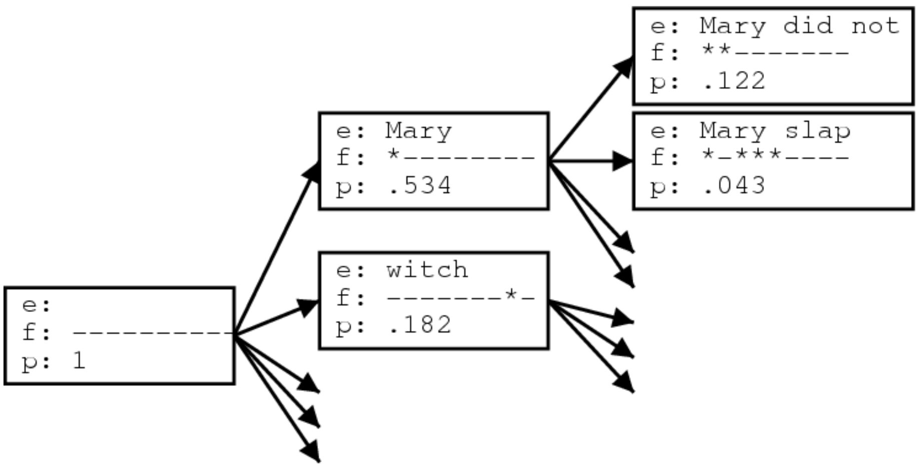

Figure 5: Decoding of *Maria no daba una bofetada a la bruja verde* (Koehn, 2016)

point of view, this is essentially a search problem, but the search space is so big (unless a maximum length is defined, infinite and, even if it is, extremely big given all the possible word combinations one could potentially generate) that it is unfeasible to compute the exact solution. In fact, Knight (1999) proved that exact SMT decoding is an NP-complete problem. For that reason, a heuristic search is performed instead, creating the output text phrase by phrase until the input text is fully covered. More concretely, a beam search is usually performed, which explores the state space level by level expanding a predefined number of nodes each time. Figure 5 shows an example of one such decoding.

2.3.1 Log-linear models

Even though phrase-based SMT was initially based on the noisy channel model, this has been later replaced by log-linear models that combine several independent models known as feature functions and directly search for the translation $\hat{E}$ with the highest $P(E|F)$ probability:

$$\hat{E} = \arg\max_{E} P(E|F) = \arg\max_{E} \prod_{i} h_i\left(E,F\right)^{\lambda_i} = \arg\max_{E} \sum_{i} \lambda_i \log h_i\left(E,F\right)$$

where $h_i\left(E,F\right)$ denotes the ith feature function and λ_i the weight assigned to it. In practice, the translation and language models coming from the original noisy channel approach are still the most relevant factors, but the architecture allows to incorporate an arbitrary number of feature functions that model different aspects of the translation process. A standard set of feature functions would include the following:

- The **language model** $P(E)$ seen above.

- The **forward and inverse translation probabilities** $\phi(\bar{e}|\bar{f})$ and $\phi(\bar{f}|\bar{e})$ seen above. The original noisy channel model only made use of the inverse translation probability, but the forward probability is also incorporated in log-linear models.

- The **forward and inverse lexical weighting** $p_w(\bar{e}|\bar{f})$ and $p_w(\bar{f}|\bar{e})$. These come directly from the word translation probabilities of the word alignments, and they can therefore be seen as an inheritance from the original word-based SMT systems, having a smoothing effect over the phrase translation probabilities:

$$p_w(\bar{f}|\bar{e}) = \max_a \prod_{j=1}^{J} \frac{1}{|\{i|(i,j) \in a\}|} \sum_{\forall (i,j) \in a} p(f_j|e_i)$$

- The **distortion probability** or distance-based reordering model, also seen above.

- A constant **word penalty**.

- A constant **phrase penalty**.

- A constant **unknown word penalty**.

- **Lexical reordering** features.

Finally, the weights λ_i of log-linear models are usually tuned to optimize the translation quality in a development set according to some automatic metric like BLEU (see Section 2.5 for more details on these). A stochastic method known as minimum error rate training (MERT) is typically used for that (Och, 2003).

2.4 Addressing the limitations of statistical machine translation

Phrase-based systems have long been the state-of-the-art in machine translation. However, these models still suffer from important limitations, which are mostly related to the following factors:

- Sparsity of natural language, causing many terms or linguistic phenomena to be missing or poorly represented in the training corpora with the inherent difficulty to properly model them, a problem that is accentuated by the scarcity of bilingual resources for the vast majority of language pairs.

- Locality of the statistical methods used, with the inherent difficulty to properly model context information, long distance dependencies and morphosyntactically diverging language pairs.

- Inflexibility of the statistical models used, with the inherent difficulty to incorporate external linguistic information from other natural language processing tools and resources.

Recent research on machine translation has mostly focused on addressing these issues, and several extensions of the phrase-based approach have been proposed with different degrees of success. This section discusses the most relevant ones as introduced in Section 2.1.2.2: factored models, tree-based models and deep learning approaches including neural machine translation. Later on, our work proposes two additional approaches to mitigate the limitations of phrase-based systems: the use of logistic regression for dynamic phrase translation probability modeling, incorporating additional lexical, context and distributional semantic features (Chapters 5 and 6), and the use of bilingual word embeddings for phrase translation similarity scoring (Chapter 8).

2.4.1 Factored models

One important limitation of traditional phrase-based systems is that they work with the surface form of words, making it difficult to incorporate linguistic information into the translation process. This becomes obvious for morphologically rich languages like Basque, where a single lemma can have dozens of different surface forms, severely accentuating the sparsity problem of all statistical approaches in natural language processing. For instance, there might be enough evidence in the training data to learn how to translate a given lemma and a given grammatical case independently, but if their corresponding surface form is not found in the parallel corpus, a traditional phrase-based system would not be able to learn its translation.

Factored models (Koehn and Hoang, 2007) try to overcome this issue by incorporating translation and generation steps for different layers of linguistic information known as factors into the log-linear model discussed in Section 2.3.1. Each translation step learns how to map some factors from one language to the other and, in that regard, they are equivalent to the models discussed for phrase-based systems, except that they are not necessarily limited to surface forms but can also work with other factors like lemmas or part-of-speech tags, for instance. At the same time, generation steps model the probability of obtaining one factor given some others. In addition to them, language models can also be built for specific factors. Figure 6 shows an example factored model with one translation step for lemmas, another translation step for the part-of-speech and morphological tags, and one generative model from the lemma, part-of-speech and morphology to the surface form.

2.4.2 Tree-based models

Another limitation of traditional phrase-based systems is that they directly map phrases from one language to the other, without considering any structural correspondence. Tree-based models (Chiang, 2005) overcome this by means of grammar rules, which incorporate variables (known as non-terminals) in addition to words (known as terminals). This way, just as a phrase-based model could have an entry in the phrase table with its associated weights for "I have not seen it → no lo he visto" and "I have not read it → no lo he leído", a tree-based model might learn how to generalize this and have an entry for "I have not

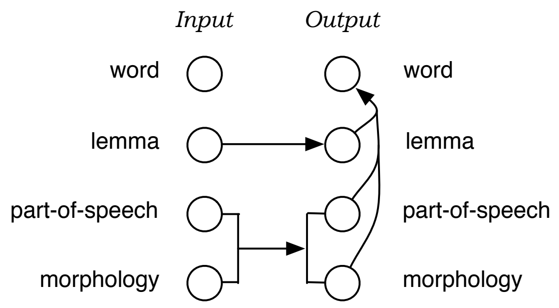

Figure 6: An example factored model (Koehn and Hoang, 2007)

X it → no lo he X”, where X is a non-terminal. During decoding, the non-terminal X will be expanded through other grammar rules following a tree structure, which is why these models are said to be tree-based. These grammar rules can be automatically extracted from word aligned corpora just as phrases, making use of syntactic annotations or not. As a consequence, the tree structures might be linguistically motivated, in which case tree-based models are commonly known as syntactic models, or not, in which case tree-based models are commonly known as hierarchical models.

2.4.3 Deep learning and neural machine translation

As discussed later in Section 3.4.3, artificial neural networks are a family of models inspired by biological neural networks (primarily the brain) that approximate unknown functions through units connected by weights, analogous to neurons. In the last few years, these models are revolutionizing artificial intelligence thanks to deep learning, which uses several such processing layers to model high level abstractions. They have achieved a dramatic improvement in the state of the art of several fields like speech recognition (Hinton et al., 2012) and image processing (Krizhevsky et al., 2012), and natural language processing is expected to be one of the next areas were deep learning will make a large impact in the near future (LeCun et al., 2015).

One of the main strengths of deep learning when compared to existing statistical methods is the use of continuous representations of words in close relation to word embeddings (see Section 4.1), which serves to mitigate the sparsity problem in natural language processing. Based on this, deep learning models have been successfully applied at the different components of phrase-based SMT systems, including the language model (Vaswani et al., 2013), the reordering model (Li et al., 2013) and the translation model (Zhang et al., 2014; Su et al., 2015; Cho et al., 2014) (see Section 4.4).

At the same time, a completely new approach known as neural machine translation has recently emerged, which uses artificial neural networks to model the translation process from end to end (Bahdanau et al., 2015; Jean et al., 2015; Sutskever et al., 2014; Luong et al., 2014). Even if it is still too early to speak about a clear improvement with respect to existing statistical systems, these models have achieved very competitive results in a surprisingly short period of time, and there is a high expectation that they could achieve a large improvement in the state of the art in the future.

2.5 Evaluation in machine translation

Unlike other tasks where simple metrics like the accuracy or the f-score can be naturally used to measure the quality of a system, evaluation is a very difficult task in MT. Broadly speaking, two different dimensions are considered when evaluating an MT system: the fidelity (i.e. how accurately the information in the input text is preserved in the output text) and the fluency (i.e. how natural and correct the output text is in the target language). As it can be seen, these are tightly related to the translation and language models of SMT systems, respectively.

Manual evaluation uses human raters to assess the quality of MT systems. One possibility is to ask the evaluators to score the system according to different aspects of the above dimensions (e.g. the clarity, naturalness, style, adequacy or informativeness of the translations). It is also possible to compare the output of different systems instead of evaluating them independently. An extrinsic evaluation can also be done, where end-users are asked to use the MT system within its intended application and their experience evaluated.

One advantage of manual evaluation is that it does not only assign numeric scores, but also serves to get insight about the behavior of each system, helping to identify the type of errors they do so that these can be properly addressed. It is also argued the best evaluation will necessarily come from humans, as we are the intended users of MT and also the ones with the linguistic knowledge to properly assess it. However, manual evaluation is also inherently subjective and costly to carry out. As a consequence, the results of a manual evaluation will always be difficult to reproduce accurately, and performing a new manual evaluation for every iteration while developing a system will likely be unpractical.

Automatic evaluation overcomes these issues by means of different heuristic methods to compare the output of an MT system with a reference translation made by a person. The most widely used evaluation metric in this regard is the bilingual evaluation understudy or BLEU (Papineni et al., 2002), which measures the similarity between the MT output and the reference translation as follows:

$$BLEU = BP \times \exp\left(\frac{1}{N}\sum_{n=1}^{N}\log p_n\right)$$

where BP is a brevity penalty defined as follows:

$$BP = \begin{cases} 1 & c > r \\ e^{(1-r/c)} & c \leq r \end{cases}$$

where c is the length of the MT output, r is the length of the reference translation, and p_n

is the modified n-gram precision score defined as follows:

$$p_n = \frac{\sum_{C \in \text{Candidates}} \sum_{\text{n-gram} \in C} \text{count}_{clip}\,(\text{n-gram})}{\sum_{C' \in \text{Candidates}} \sum_{\text{n-gram}' \in C'} \text{count}\,(\text{n-gram}')}$$

2.6 Moses

Moses (Koehn et al., 2007; Koehn, 2016) is an open source toolkit for statistical machine translation. It is licensed under the GNU Lesser General Public License (LGPL) and its development is mainly supported by the European Union. The toolkit provides tools to train and tune SMT systems as well as a decoder to run them, and in addition to the dominant phrase-based paradigm it also supports tree-based and factored models. Thanks to its active development community, ease of use, wide set of supported features and state-of-the art performance, Moses has become the de facto benchmark for SMT research, and it is also widely used in commercial systems.

In order to word align the parallel training corpus, Moses is commonly used together with GIZA++ (Och and Ney, 2003), an extension of the GIZA word aligner developed by Franz Josef Och and licensed under the GNU General Public License (GPL). It implements the IBM Models 1-5 and HMM models discussed in Section 2.2 with several extensions, improvements and optimizations. Apart from the official version, Qin Gao implemented a multi-threaded version of GIZA++ called MGIZA, which is also integrated into Moses.

As for the language model, Moses supports the SRI, IRST, RandLM, KenLM, DALM, OxLM and NPLM toolkits. They are all open source projects with the exception of SRI, which is nonetheless freely available for research purposes. Moses includes KenLM by default, but the rest of the toolkits are also well integrated and can be easily used instead if preferred.

2.7 Conclusions

From rule-based to corpus-based systems, there are many different approaches to machine translation, phrase-based SMT being the dominant paradigm in recent times. In spite of its success, however, this approach still suffers from important limitations like the sparsity of natural language, the locality of the statistical methods it uses, and the difficulty to incorporate external linguistic information. In this work, we try to address these issues using two different strategies. First, in Chapter 5 we propose the use of logistic regression for dynamic phrase translation probability modeling, which we prove to be a generalization of the standard translation probabilities used in phrase-based SMT. This provides an alternative to factored models to incorporate linguistic information, as we do with distributional semantic features in Chapter 6, with the advantage of allowing context features to overcome the locality problem as well as additional features shared across different phrase pairs to overcome the sparsity problem. Second, in Chapter 8 we explore the use of bilingual word embeddings, like the ones we propose in Chapter 7, for phrase translation similarity scoring, incorporating distributional semantic knowledge into the translation model while addressing the sparsity problem thanks to the continuous word representations, in close relation to deep learning models for machine translation. Apart from that, we have seen that, given the cost and subjectiveness of manual evaluation, automatic metrics are typically used instead, as we do in this project with BLEU, the most widely used one among them. Finally, we have presented Moses, which has become the de facto toolkit for SMT research and also the one we use for our work.

3 Large scale machine learning

Machine learning is a subfield of computer science that studies algorithms that learn to make predictions or decisions based on data. There are many different tasks that fall under this broad definition, which are commonly classified into **supervised learning**, where the data contains the characterization of each training instance along with its expected output, **unsupervised learning**, where the latter is not available so the algorithm has to find the structure of the data, **semi-supervised learning**, where only a subset of the training instances contain the expected output, and **reinforcement learning**, where the algorithm receives feedback about its performance dynamically.

This chapter focuses on supervised learning, which is the most widely used one to the extent that machine learning is sometimes used as a synonym for it, and it is also the one that is relevant for this work. Section 3.1 first presents the basic concepts in this area. Section 3.2 then introduces a family of supervised learning algorithms called linear models and discusses a very simple model that belongs to this group known as the perceptron. Section 3.3 presents a more sophisticated linear model named logistic regression together with gradient descent, a widely used optimization procedure for this and other models. The limitations of these linear models are then discussed in Section 3.4 together with different approaches to overcome them. Section 3.5 then describes the peculiarities and difficulties of applying these learning algorithms to large amounts of data. After that, Section 3.6 presents Vowpal Wabbit, a widely used open-source library for such large scale learning problems, and also the one we use in this project. Section 3.7 concludes the chapter.

In addition to the specific citations in the text, Goodfellow et al. (2016), Leskovec et al. (2014), Jurafsky and Martin (2008) and the online courses "Machine Learning"[1] by Andrew Ng and "Neural Networks for Machine Learning"[2] by Geoffrey Hinton have been used as a general reference for writing this chapter.

3.1 Basic concepts in supervised learning

Following the definition above, the goal of supervised learning is to, given a **training set** of m different (x, y) pairs, where x is is the feature vector that characterizes each instance and y is the correct output that corresponds to it, learn a function to predict the output y' of any new, possibly unseen instance x'. Each feature x_i and the output y can be either **categorical** if its value is taken from a discrete set, or **numerical** if its value is an integer or a real number. For the sake of simplicity, in this work we will follow a standard approach to convert categorical features into numerical ones as follows: for each value that a categorical feature can take, we will create a new numeric feature that will be set to 1 for instances that took its associated value for the original categorical feature, and 0 for the rest. For instance, for a categorical feature corresponding to the set {noun, adjective, verb, adverb} we would create 4 new numerical features (one for each value in the set) so

[1]https://www.coursera.org/learn/machine-learning/
[2]https://www.coursera.org/course/neuralnets

that the one corresponding to "noun" would be set to 1 for nouns and 0 for adjectives, verbs and adverbs, the one corresponding to "adjective" would be set to 1 for adjectives and 0 for nouns, verbs and adverbs etc. As for the output y, supervised learning problems whose output is categorical are called **classification** problems, and those whose output is numerical are called **regression** problems. The special case of the former with only two possible output values is called **binary classification**.

Needless to say, a good classifier or regressor will be one that gives the best possible predictions for new instances. For that reason, a **test set** is usually kept apart from training and used to evaluate the performance of different systems according to task relevant measures (e.g. accuracy for classification or mean squared error for regression). During training, each supervised learning algorithm will learn a model to make such predictions based on the regularities it observes in the training set. It might therefore seem obvious that the performance of the model in the training set should improve as the training goes on. However, it might happen that, as the model learns to perform better in the training set, it starts considering regularities there that are not such in reality (e.g. due to the sampling noise) and, by doing so, it looses its generalization capacity, yielding to worse results in the test set. This problem is known as **overfitting**. The opposite can also happen, as a model might miss relevant regularities in the training set due to the assumptions it makes, yielding to worse results not only in the training set, but also in the test set. This other problem is known as **underfitting**. Needless to say, a good model should find the medium term between them. This is called the **bias-variance tradeoff**, where **bias** refers to the error from the assumptions made by the model, and **variance** refers to the error from the small fluctuations in the data. By adjusting different parameters of the model or the training algorithm and seeing its effect in the test set, or by observing the error in the training and the test sets to decide when to stop the training, a better and better tradeoff could be found. However, this would not be methodologically acceptable, as there would be a clear risk to start overfitting the test set itself, so the error on it would not be representative anymore. For that reason, an additional development or **validation set** is typically used for that. This validation set is kept apart from the training set and used to tune all these parameters, and it is only after the definitive values for them are chosen that the test set is used to report the performance of the model.

Another important distinction in machine learning methods is that of online learning as opposed to batch learning. In offline learning or **batch learning**, the entire training set is available from the very beginning and processed at once to create a model. The opposite approach is **online learning**, where the training data is not processed altogether, but becomes available sequentially, so the model keeps adjusting itself to the instances it sees each time on the fly. This makes it possible to have dynamic models that adjust themselves to continuously changing environments. For instance, spammers would try to exploit new attack vectors as the old ones are mitigated, so an online classifier for spam filtering could adapt itself to detect these new practices. In addition to this, as discussed later in Section 3.5 processing all the data together might not be computationally feasible for very large training sets like the ones we have for SMT, so online learning is often the most suitable approach to large scale machine learning.

3.2 Linear models and the perceptron

Linear models are a family of supervised learning models that take a linear combination z of the input features

$$z(x) = \sum_i w_i x_i + b$$

and predict their output as a function of this linear combination:

$$h(x) = f(z(x))$$

In the formulation above, $h(x)$ refers to the prediction or hypothesis made by the model for the input x, w_i denotes the **weight** associated to each input feature x_i, and b is the so called **bias term**. An alternative way of dealing with this bias term is to take an additional input feature x_0 that will always have a constant value of 1, and treat the bias term as its corresponding weight w_0. In any case, the parameters of a linear model are precisely the weights and the bias term, and the training will thus consist in finding the optimal values for them based on the instances in the training set.

Depending on the output function, the optimization objective and the training procedure, different linear models are defined. The **perceptron** is arguably the simplest one that one could conceive, and we will therefore use it for illustrative purposes throughout this section. The perceptron is used for binary classification, and its output simply depends on the sign of the linear combination z:

$$h(x) = \begin{cases} 1 & z(x) > 0 \\ 0 & z(x) \leq 0 \end{cases}$$

In this notation, 1 and 0 are simply the identifiers for the predicted class, so their exact meaning will depend on the definition of the problem (e.g. 1 might mean true and 0 false).

From the graphical point of view, this is equivalent to drawing a straight line known as the **decision boundary** in a 2-dimensional feature space, so that everything that lays in one side of the line will be predicted to belong to one class and everything that lays in the other side will be predicted to belong to the other class. For higher dimensional feature spaces, the straight line is simply generalized to a hyperplane.

Figure 7 shows an example of this for a toy problem where the sex of a person has to be predicted given their age and the fundamental frequency of their voice (the physical magnitude associated with the pitch). As it is well known, men have a lower fundamental frequency than women, but the age is also a relevant factor, as children, for instance, have a higher fundamental frequency. In this artificial example, both classes are found in different regions in the training set, and a meaningful decision boundary is accordingly defined.

Needless to say, this decision boundary (that is, the exact straight line to draw) will be defined by the parameters of the model, which were the weights and the bias term. So, moving from the mathematical formulation to the graphical representation, training the classifier will consist in finding the straight line that best separates the set of examples in the training set.

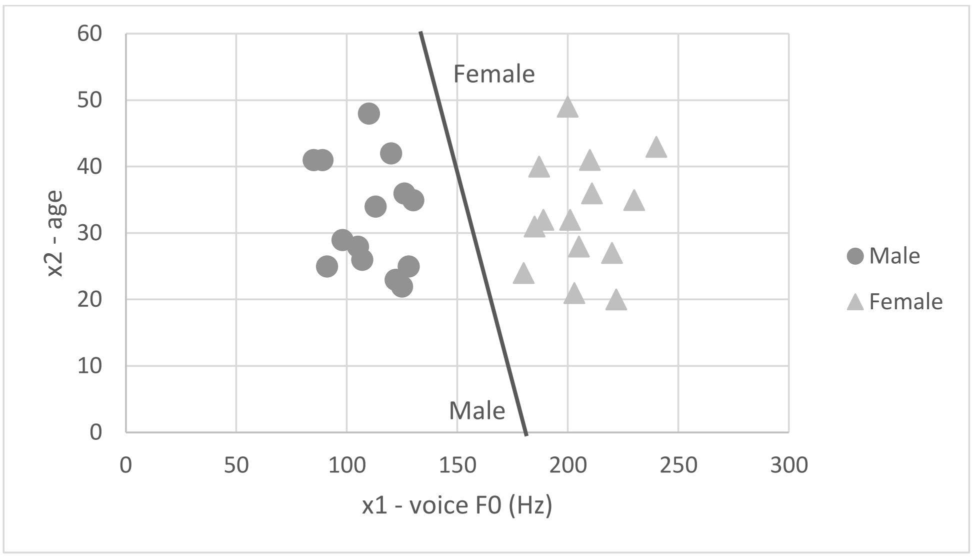

Figure 7: Example decision boundary for a linearly separable problem

The perceptron learning algorithm provides a very simple training procedure that is guaranteed to find the decision boundary that perfectly separates both classes in the training set as long as this decision boundary exists. When that condition holds, as it is the case of the example in Figure 7, the training data is said to be **linearly separable**.

For that purpose, the perceptron learning algorithm follows an iterative process. It treats the bias term as an additional weight as discussed before, and starts with a fixed weight vector, typically initialized to all zero. It then picks the instances in the training set one by one, and predicts its class by using the current weight vector. If the predicted class is 0 and the real class is 1, the feature vector is added to the weight vector; if the predicted class is 1 and the real class is 0, the feature vector is subtracted from the weight vector; and if the prediction is correct, the weight vector is kept unchanged.

This surprisingly simple learning algorithm is guaranteed to converge as long as the training set is linearly separable. In other words, the perceptron learning algorithm is guaranteed to eventually find the straight line that separates the classes in the training set as long as this straight line exists.

3.3 Logistic regression and gradient descent

As discussed in Section 3.2, the perceptron is a very simple model for binary classification, but also a rather limited one. One way to build more powerful models is to use a more sophisticated output function in a linear model. In the case of **logistic regression**, the so called **sigmoid or logistic function** is used for that purpose:

$$h(x) = \frac{1}{1 + e^{-z(x)}}$$

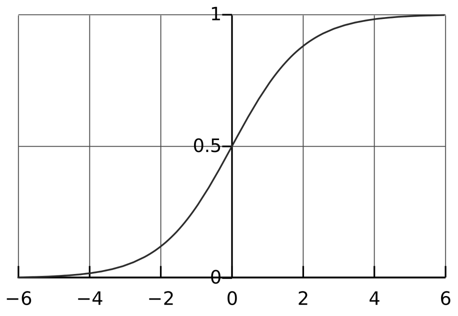

Figure 8: The sigmoid or logistic function

In this case, the hypothesis $h(x)$ is interpreted as the probability of a given case belonging to the positive class 1, that is, logistic regression will predict $p(1|x) = h(x)$ and $p(0|x) = 1-h(x)$. This is in contrast to what perceptrons do, which make a hard prediction instead of giving a probability distribution.

In order to understand why it is possible to interpret the output of logistic regression as a probability, is should be observed that the sigmoid function is a smooth curve that always takes a value between 0 and 1 as shown in Figure 8.

When training a logistic regression classifier, the so called **conditional maximum likelihood estimation** is used. In other words, logistic regression tries to find the set of weights $\hat{w}$ that maximizes the probability of observing the classes in the training set given their corresponding features. This can be formulated as follows, where $y^{(i)}$ stands for the observed class of the ith training instance, $x^{(i)}$ for its corresponding feature vector, and $p(y^{(i)}|x^{(i)})$ is the probability predicted by the classifier for this particular instance and the weight vector w:

$$\hat{w} = \arg\max_{w} \prod_{i=1}^{m} p(y^{(i)}|x^{(i)})$$

Taking advantage of the properties of the logarithm and the fact that, in binary classification, the observed class $y^{(i)}$ will either be 0 or 1, this can be reformulated as follows:

$$\begin{aligned}\hat{w} &= \arg\max_{w} \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}) = \arg\max_{w} \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}) \\ &= \arg\max_{w} \sum_{i=1}^{m} \left(y^{(i)} \log h\left(x^{(i)}\right) + \left(1-y^{(i)}\right) \log\left(1-h\left(x^{(i)}\right)\right)\right) \\ &= \arg\min_{w} -\sum_{i=1}^{m} \left(y^{(i)} \log h\left(x^{(i)}\right) + \left(1-y^{(i)}\right) \log\left(1-h\left(x^{(i)}\right)\right)\right)\end{aligned}$$

Based on this last expression, an **error function** E can be defined so that $\hat{w} =$

$\arg\min_w E$:

$$E = -\sum_{i=1}^{m} \left(y^{(i)} \log h\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h\left(x^{(i)}\right)\right)\right)$$

Intuitively, E measures the error made by the model when predicting the class of the instances in the training set, and the goal of the training algorithm is thus to find the set of weights that minimize this error. In order to prevent the overfitting problem described in Section 3.1, a **regularization term** $R(w)$ that depends on the weight vector is often added to the error function. The purpose of regularization is to penalize extreme values for the weights that might make the model fit better to the training set but are likely to harm its generalization ability. The so called L1 and L2 regularization are the most widely used ones, which are defined as follows:

$$R_{L1}(w) = \sum_i |w_i|$$

$$R_{L2}(w) = \sum_i w_i^2$$

Independently to the exact function used, the regularization term is added to the error seen above according to some factor λ, so the error function is redefined as follows:

$$E = -\sum_{i=1}^{m} \left(y^{(i)} \log h\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h\left(x^{(i)}\right)\right)\right) + \lambda R(w)$$

No matter if regularization is used or not and what function is used for it, training a logistic regression classifier is hence reduced to solving the particular optimization problem of minimizing this error function. Even though it is mathematically possible to calculate the optimal solution analytically, this approach does not scale well, so a numerical method is typically used instead.

The **gradient descent** algorithm is one of the most commonly used ones. It follows an iterative process were, at each step, each weight is updated depending on the derivative of the error function with respect to it according to a **learning rate** ϵ. These partial derivatives correspond to the following expression in the case of logistic regression:

$$\frac{\partial E}{\partial w_j} = -\sum_{i=1}^{m} \left(y^{(i)} - h\left(x^{(i)}\right)\right) x_j^{(i)}$$

This way, gradient descent will update the weights at each iteration as follows:

$$\Delta w_j = -\epsilon \frac{\partial E}{\partial w_j} = \sum_{i=1}^{m} \epsilon \left(y^{(i)} - h\left(x^{(i)}\right)\right) x_j^{(i)}$$

Under certain general assumptions (most notably, the function to optimize has to be convex), this algorithm is guaranteed to converge to a local optimum for a small enough

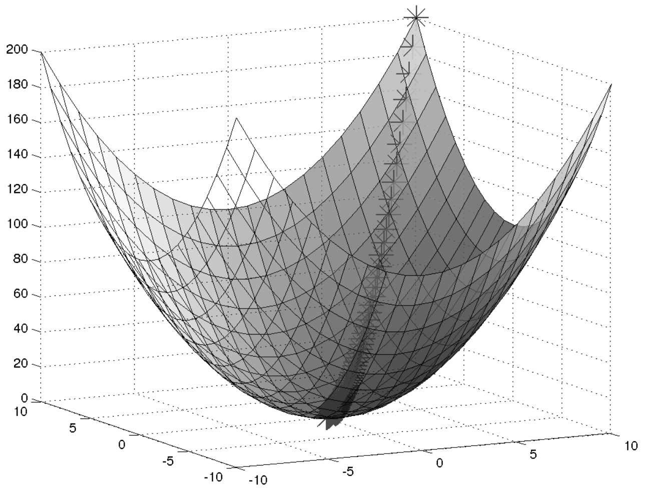

Figure 9: Visualization of gradient descent in a 2-dimensional weight space

learning rate. In the case of logistic regression, there is a single minimum, so the optimal solution found by gradient descent will also be the global optimum.

The behavior of gradient descent is better understood if the error surface is graphically represented in the weight space as shown in Figure 9. Once again, the example corresponds to a 2-dimensional weight space so that it can be properly visualized, but this can be generalized to a hyperplane for higher dimensional spaces. As discussed before, the goal of the training phase is to find the minimum of this error function or, from the graphical point of view, the coordinates defined by the weights for which the error is minimum. The gradient descent algorithm starts from a random point in this error surface and, in each iteration, moves in small steps in the direction of steepest descent. Considering that the derivative is geometrically interpreted as the slope of a curve at a given point, it is easy to establish the relation between the mathematical formulation above and the geometrical interpretation here.

3.4 Addressing the limitations of linear models

The linearity assumption that is inherent to the very same definition of linear models can pose an important limitation on what these models can learn. Consider, for instance, the example in Figure 10, which shows another possible training set for the toy problem of predicting the sex of a person discussed earlier. It can be clearly seen that, in this other configuration, it is not possible to draw a straight line that correctly separates both classes in the training set. In other words, the training set is not linearly separable, so linear classifiers like the perceptron or logistic regression discussed earlier will perform poorly. Nevertheless, it is obvious that the problem here is not in the training set itself, as it would

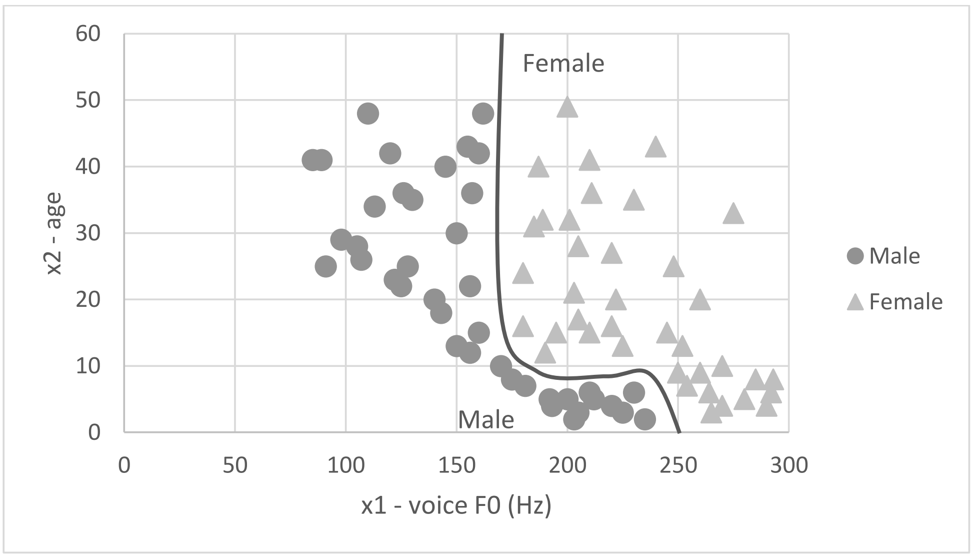

Figure 10: Example decision boundary for a problem that is not linearly separable

be perfectly possible to correctly separate both classes with a curve as shown in Figure 10. Therefore, it is the linear models themselves that are not expressive or powerful enough to learn to discriminate between the two classes in this case.

In this section, some common approaches to address these limitations are described. Section 3.4.1 first discusses feature engineering, the approach of manually designing additional features to improve the performance of machine learning algorithms, which we use in this project for the phrase translation probability scoring logistic regression model we propose. Support vector machines are then introduced in Section 3.4.2, which are linear models that are nevertheless able to learn non-linear decision boundaries in the original feature space thanks to the kernel trick. Finally, Section 3.4.3 discusses artificial neural networks and deep learning, which can be seen as a non-linear extension of the models seen so far and are typically used to train word embeddings, presented later in Section 4.1 and extensively used throughout the rest of the project.

3.4.1 Feature engineering

Feature engineering is the process of manually building a right set of features so that a given learning algorithm can model the problem as good as possible. For instance, for the example shown in Figure 10 one could create new features to make the problem linearly separable and then apply a linear classifier like logistic regression or the perceptron. This has traditionally been one of the main focuses in applied machine learning to the extent that the amount of data available and the feature set used have often been considered the key for achieving good results, more than the learning algorithm itself.

A typical approach would be to use domain knowledge to design a set of features that are believed to be good for modeling the problem. Although this might look to go against the automation nature of machine learning, it should be noted that machine learning is most useful when we are not able to come up with an algorithm to explicitly solve a particular problem, which does not necessarily mean that we know nothing about how to solve the problem. For instance, it would be extremely difficult for us to design a step-by-step method that reliably tells whether a given email message is spam or not, but we could easily think of features that could be useful for this task (e.g. the presence of certain words, the email address of the sender, whether the email contains the name of the receiver...).

However, this is not limited in any case to designing features that add new and relevant information based on domain knowledge. For instance, one could create a new feature by simply **scaling** an existing one (e.g. taking its logarithm or square, centering it around 0...). In spite of its simplicity, most learning algorithms greatly benefit from this feature normalization.

More interestingly, one could also create new features that combine existing ones (e.g. taking the product of two existing features). These **interaction features** are particularly relevant for linear models, as they only have a fixed weight for each feature that is independent from the rest. The underlying issue becomes obvious with the paradigmatic example of the XOR problem. The XOR logical operator takes two binary arguments and returns 0 if both are equal (i.e. 0, 0 or 1, 1) and 1 if both are different (i.e. 0, 1 or 1, 0). As proved by Minsky and Papert (1969), it can be easily seen that this frustratingly simple problem is not linearly separable, and linear models therefore fail to properly model it. This happens because the output of XOR is given by the relationship between the two input features and, as said before, linear models can only weight them independently. However, this can be easily solved by adding an interaction feature for the original input features.

Needless to say, these interaction features do not necessarily have to be manually designed, as one could for instance take every quadratic or cubic combination of all the input features. However, depending on the number of features this can easily lead to a combinatorial explosion, known as **feature explosion**, increasing the computational complexity of training the model and greatly increasing the risk of overfitting.

3.4.2 Support vector machines

At the most basic level, **support vector machines** (SVM) are linear models known as **large margin classifiers**, since they choose the linear decision boundary that maximizes the margin between the instances of both classes in the training set. For that purpose, they use exactly the same output function as perceptrons:

$$h(x) = \begin{cases} 1 & z(x) > 0 \\ 0 & z(x) \leq 0 \end{cases}$$

and train the model to minimize the following error function, which is known as the **hinge loss**:

$$E = \sum_{i=1}^{m} \max\left(0, 1 - y^{(i)} z(x^{(i)}) + \left(1 - y^{(i)}\right) z(x^{(i)})\right) + \lambda R(w)$$

The intuition behind the hinge loss is that it does not only penalize wrong predictions, but also pushes $z(x)$ to large values for correct prediction, as $|z(x^{(i)})| \geq 1$ must hold for the error of the ith instance to be 0 even if its class is correctly predicted.

However, the real power of SVM comes with the so called **kernel trick**, which is a computationally efficient way that SVM allows to create implicit features based on the distance to the training instances. For that purpose, different kernel functions with different parameters are defined, as it is the case of polynomial kernels or the Gaussian radial basis function kernel. A linear kernel corresponds to the basic model discussed above. This way, thanks to the kernel trick the data is projected to a higher-dimensional feature space and SVM behaves as a linear model there, making it possible to learn non-linear decision boundaries in the original feature space. For this reason, SVM is often said to be the most powerful black box classifier.

3.4.3 Artificial neural networks and deep learning

In spite of their simplicity and limitations, it is interesting to see how similar linear models are from the basic building blocks of our brain: neurons. At the most basic level, a neuron receives some electrical input from the dendritic tree and, when this input is high enough, it activates its output through the axon. Even though the actual biochemical process is far more complex, linear models can therefore be taken as ideal models of neurons. When doing so, the sigmoid function is often used as the activation function just as seen in logistic regression. But, needless to say, our intelligence does not come from what neurons can do on their own but what they can do when connected together. This can be taken as inspiration to connect these ideal models of neurons to form more complex networks, and this is precisely how artificial neural networks are defined.

More precisely, an **artificial neural network** (ANN) is formed by a set of units connected with some weights that each take some input features, calculate a linear combination of them according to the weights, apply an activation function, and output this value. Depending on the schema that these connections follow, different ANN architectures are defined. For instance, **feedforward neural networks**, which are the most simple and also one of the most widely used ones, are formed by an input layer, an output layer, and one or several hidden layers that are each connected to units in subsequent layers (i.e. without forming directed cycles). The graph in Figure 11 illustrates this basic schema for a network with a single hidden layer.

As it can be seen, the input and the output layers are not new from linear models. In fact, a network without any hidden unit would be formed by a single neuron, which is indeed a linear model. As for the hidden layers, a way of seeing them is that hidden units are linear models that each learn a new feature that will then be useful to make the final prediction. For instance, in our previous example where we try to predict the sex of a person given their age and the fundamental frequency of their voice, a hidden unit might learn to output whether the input corresponds to a child or not, which might then be useful to predict the sex of the person together with other features. However, in contrast with the idea of feature engineering developed in Section 3.4.1, these new features are learned

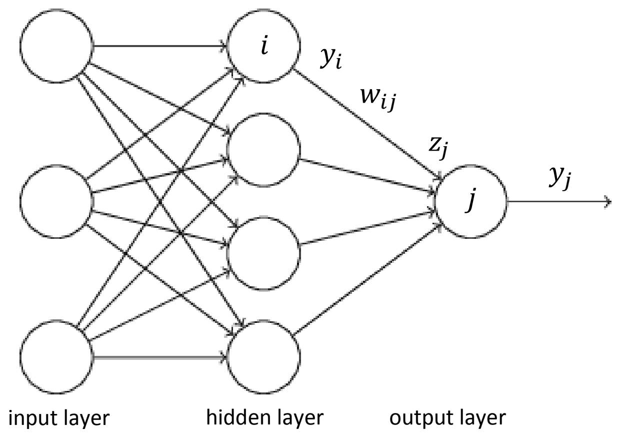

Figure 11: A feedforward neural network with a single hidden layer

automatically by the network, and that is precisely why they are called hidden, as we do not really control what they are doing. Thanks to this flexibility, ANNs are able to learn non-linear decision boundaries. For instance, for the training set in Figure 10, for which linear models could not find an appropriate decision boundary, an ANN might be able to learn a decision boundary like the one in the chart.

At the same time, by adding more than one hidden layer to the network, it would be able to learn more and more abstract (or, in other words, deeper) representations of the input data that could then be helpful to correctly predict the output, and this is how **deep neural networks** (DNN) are defined. DNNs belong to the **deep learning** branch of machine learning, a broader term that encompasses other approaches that make use of multiple processing layers to obtain high-level abstractions of the input data (LeCun et al., 2015).

No matter what architecture is used, though, an ANN is parametrized by its weight just as linear models (there is also the bias term for each unit but, as seen before, those can be seen as additional weights for a constant one-value input). Therefore, training will again consist in finding the right values of these weights for the given training set. For that purpose, an error function E is defined on the training set according to the desired output, and the gradient descent algorithm is then typically used to iteratively find the optimal set of weights that minimize this error function, just as seen for logistic regression in Section 3.3. The update value for each weight in each iteration of gradient descent will again be given by the derivative of the error function with respect to it multiplied by the learning rate ϵ:

$$\Delta w_{ij} = -\epsilon \frac{\partial E}{\partial w_{ij}}$$

In order to calculate that value, we will first calculate the derivative of the error function with respect to the total input received by the unit z_j using the chaining rule and assuming a sigmoid activation function:

$$\frac{\partial E}{\partial z_j} = \frac{\partial y_j}{\partial z_j}\frac{\partial E}{\partial y_j} = y_j\left(1-y_j\right)\frac{\partial E}{\partial y_j}$$

Having done that, the chaining rule can again be applied to calculate the derivative of the error function with respect to the weight w_{ij}:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}}\frac{\partial E}{\partial z_j} = y_i\frac{\partial E}{\partial z_j}$$

This way, we obtain the derivative of the error function with respect to the weights in the last layer, which we can use to update them. However, this is not directly applicable to the weights in previous layers since, in the case of a hidden unit, its expected output is not known a priori (that is precisely why they are called hidden). However, making use of the chaining rule, it is possible to calculate the derivative of the error function with respect to the output of each of the units in the previous layer:

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial z_j}{\partial y_i}\frac{\partial E}{\partial z_j} = \sum_j w_{ij}\frac{\partial E}{\partial z_j}$$

Based on that, the same procedure seen before for the weights in the last layer can be applied to obtain the update value for the ones in the previous layer. All in all, starting with the derivative of the error function with respect to the output of the ANN the error is backward propagated to the units in the previous layers. For that reason, this method is known as the **backpropagation algorithm**.

From the graphical viewpoint, the behavior of this procedure is essentially the same as the one discussed for logistic regression in Section 3.3, with the difference that the error surface has a more complex shape and will generally have more than one local minima. The training will start in a random point in the weight space, and will advance step by step in the direction of steepest descent until it reaches one of these minima.

3.5 Large scale online learning

In principle, every technique discussed throughout the chapter can be applied regardless of how big the training set is. In fact, having a large dataset is always desirable, as it reduces the sampling noise and is therefore helpful to improve the variance of a model without affecting its bias (or, in other words, it reduces the risk of overfitting without affecting the generalization capacity of the model).

However, several practical issues arise when trying to apply the techniques discussed so far in a very large scale. First, the variant of gradient descent presented here, sometimes referred to as **batch gradient descent**, needs to process the entire dataset to compute the update value for each weight, which has to be repeated many many times until it converges. When the training set is very big, this is likely to be too slow in practice. An alternative approach, known as **stochastic gradient descent**, is to compute the update for a single or a fixed number of training instances each time. The latter are known as mini-batches, so this variant is also called **mini-batch gradient descent**. Updates in stochastic gradient descent are more unstable, so the learning rate is usually decreased with time but, since they are also performed much more often, the training tends to converge much faster. Moreover, thanks to this variability the algorithm can potentially escape poor local optima at the beginning.

Another important aspect of stochastic gradient descent is that it is an online learning algorithm. This is relevant because, in many real large scale machine learning settings, such as the scam filtering discussed before, new training data becomes available constantly, and it is useful to have a model that keeps adapting to it. What is more, training instances can even been thrown away after they are processed saving storage space. This becomes relevant in the so called big data era where, in many real situations, the bottleneck is not the availability of the data but the capacity to process it.

Another issue related to large scale machine learning is that of **feature sparsity** and **feature vectorization**. In many scenarios, in particular in those where categorical features are converted into numerical features as discussed in Section 3.1, the feature set tends to be very large, and also very sparse (i.e. most of the features are 0 for a given instance). This is very common in natural language processing, where each instance (e.g. a sentence or a phrase) might be characterized by the words or bigrams it contains, which will only be a few out of the entire vocabulary. This poses a challenge first for vectorizing the features (i.e. organizing them into an ordered vector associated with the weight vector) and also for efficiently processing them, as it would be a waste of resources to traverse all the possible features when only a few of them take a non-zero value. In fact, in online learning scenarios the feature set might even be unbounded, as previously unseen words or bigrams are likely to occur in new training instances, for example. In order to overcome these issues, the so called **hashing trick** is often used, which converts any feature identifier, such as a word or a bigram, into an integer by means of a hashing function, which can then be used to access the weight that would correspond to it. When doing so, it is important to consider the range of the hashing function, as too big values would unnecessarily increase the space requirements and too small values could cause collisions among the features, harming performance.

Finally, parallelization is usually another desirable feature in large scale machine learning for scalability over big clusters. However, unlike batch gradient descent, in which training instances can easily be processed in parallel following for instance the standard MapReduce model, stochastic gradient descent is not easy to parallelize, as the weights have to be updated after each training instance is processed. An alternative is to run stochastic gradient descent in different nodes independently distributing the input data among them, and then merge these updates periodically using a tree-like structure over the nodes for efficient communication.

3.6 Vowpal Wabbit

Vowpal Wabbit is an efficient online learning system. Its development is led by John Langford, currently under Microsoft Research and previously under Yahoo! Research. It is written in C++ and licensed under the 3-clause BSD license, and it can be used through its command line interface or as a C++ library with bindings for Python.

Vowpal Wabbit's main strength is its efficiency along with its flexibility. It is primarily used to work with linear models and, by specifying different loss functions, linear regressors, logistic regression classifiers and SVMs with a linear kernel can be built. In addition to that, Vowpal Wabbit also supports feedforward neural networks with one hidden layer, online kernel SVM (an adaptation of the kernel trick for online learning), Latent Dirichlet Allocation, and more. It uses stochastic gradient descent with a decay learning rate as its default optimization algorithm, but it also supports other methods not covered here like BFGS and conjugate gradient.

Vowpal Wabbit also stands out for its scalability. It makes use of the hashing trick discussed in Section 3.5 for feature vectorization, and it offers an out-of-core implementation (i.e. it does not need to load all the data into memory at once in order to process it). Thanks to this, Vowpal Wabbit's memory usage is completely independent from the dataset used and its size, and depends primarily on the range selected for the hashing function. Finally, Vowpal Wabbit also offers parallel training following the schema discussed in Section 3.5 with support for Apache Hadoop.

3.7 Conclusions

In this chapter, we have presented the foundations of supervised machine learning with a focus on linear models. These linear models are often used for large scale machine learning, where the massive number of features and training instances typically make up for their modeling limitations and scalability becomes the key factor. Given the big size of the bilingual data used to train SMT systems, we also approach the dynamic phrase translation probability model we propose as a large scale machine learning problem, and adopt logistic regression with online learning, stochastic gradient descent, parallel training and the hashing trick. For that purpose, we use Vowpal Wabbit, an efficient, feature rich online learning system that is widely used for large scale machine learning. However, we have seen that linear models have important limitations, and discussed different approaches to overcome them. Among them, we use feature engineering for the dynamic phrase translation probability model we propose, whereas artificial neural networks are used to train word embeddings, presented next in Section 4.1 and extensively used throughout the rest of the project.

4 Distributional semantics

The distributional hypothesis states that the meaning of each word in a language is defined by its usage and can therefore be characterized by the words it is generally surrounded by (Harris, 1954; Firth, 1957). Based on this, distributional semantics attempts to build abstract representations of words based on their usage in a large corpus. These representations have a direct application in several computational semantics tasks such as semantic textual similarity (STS) (Agirre et al., 2015), but they have also been applied to improve the performance of many other natural language processing tools.

One approach to distributional semantics that has attracted a lot of attention in the last decade, discussed in Section 4.1, is to create dense, low-dimensional vector representations of words, which are known as word embeddings. Word clustering, presented in Section 4.2, is another approach that groups words in discrete classes according to their meaning. There have also been attempts to apply distributional semantics in cross-lingual settings. Section 4.3 discusses bilingual word embeddings, which attempt to bring these continuous representations of words in two or more languages to a shared vector space. Finally, Section 4.4 presents different approaches proposed in the literature to apply distributional semantics to improve machine translation performance, and Section 4.5 concludes the chapter.

4.1 Monolingual word embeddings

Word embeddings are dense, low-dimensional representations of words in a continuous vector space distributed in some meaningful way from the semantic point of view. In particular, the distance (e.g. the euclidean distance or the cosine distance) between the embeddings of two given words should typically correlate with their semantic similarity. This is regarded as an effective way to address the sparsity problem in natural language processing, as most systems traditionally treat words as atomic units and therefore have trouble to properly model phenomena that is not explicitly present in the training data.

Many techniques have been proposed to create such word embeddings, which have been broadly classified into two main families: count models and predict models (Baroni et al., 2014). **Count models** compute statistics of the frequency each word co-occurs with its neighbors over a large corpus, and then reduce these counts to a dense, low-dimensional vector. **Predict models** follow an alternative approach and use supervised learning techniques to predict a word given its context or the other way around, and take the underlying representation that the model creates for each word.

Early attempts to build vector space word models belong primarily to the first group and date back to late 1980s. One of the most influential ones was **Latent Semantic Analysis** (LSA), also referred to as Latent Semantic Indexing (LSI) in the field of information retrieval (Dumais et al., 1988; Deerwester et al., 1990). LSA starts with a sparse matrix, known as the occurrence matrix, where rows corresponds to words, columns corresponds to documents and each cell contains some weight for the frequency in which the word in question occurs in that document (e.g. simple counts or its tf-idf value). Once the occurrence matrix is built, dimensionality reduction techniques and, typically, singular value decomposition (SVD) is used to find a low-rank approximation of the occurrence matrix so that the number of columns is reduced to some predefined value (e.g. 100 or 300) while minimizing the reconstruction error. The rows in the resulting matrix can thus be taken as low-dimensional vector representations of their corresponding words. This basic technique later evolved giving place to more sophisticated models like Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 1999) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003).

As for the family of predict models, most techniques in this group are based on ANNs, previously discussed in Section 3.4.3. The first proposals to use ANNs to build continuous vector representations for words coincide with the development of LSA in the late 1980s (Hinton, 1986; Rumelhart et al., 1986), but did not attract as much attention. With the revived interest in ANNs in the last decade and its application in NLP, new and influential models were developed and successfully applied in a variety of tasks (Bengio et al., 2003; Collobert and Weston, 2008; Turian et al., 2010; Huang et al., 2012). However, it is the work by Mikolov et al. (2013a,c) and the public release of the **word2vec** software implementing their model that is considered a breakthrough in this regard. Unlike previous models, its efficiency made it possible to learn high quality word embeddings from a billion word corpus in a matter of hours, achieving very good results.

This work does not actually develop a single architecture but two: continuous bag-of-words (CBOW) and skip-gram, which are illustrated in Figure 12. Both models are similar in that they keep two vector representations for each word, one for when it is the central word and another one for when it is a context word, which can be seen as weights that connect the input layer to the hidden layer and the hidden layer to the output layer in a shallow feedforward neural network (i.e. each word vector would correspond to the weights that connect its corresponding input or output unit to the hidden layer). Based on this, **CBOW** tries to predict words given their neighbors in a fixed window, that is, the average of the context word vectors is used to predict the central word. **Skip-gram** follows just the opposite approach and tries to predict surrounding words in a fixed window given the central word, that is, the central word vector is used to predict each context word independently. Once the training is done, the context word vectors are simply discarded and the central word vectors are taken as word embeddings. The authors argue that CBOW offers faster training times, while skip-gram learns better representations for infrequent words.

As for the training itself, computing the cost of each instance is very expensive in the original formulation, as calculating the probability of a word given its context or vice versa takes linear time with respect to the vocabulary size. In order to address this issue, the initial approach in Mikolov et al. (2013a) used **hierarchical softmax** (Morin and Bengio, 2005), representing the vocabulary as a Huffman tree and thus reducing the complexity of computing the cost of each instance to logarithmic time with respect to the vocabulary size. The subsequent work in Mikolov et al. (2013c) proposed a simpler alternative called **negative sampling**, which is itself a simplification of the noise contrastive estimation

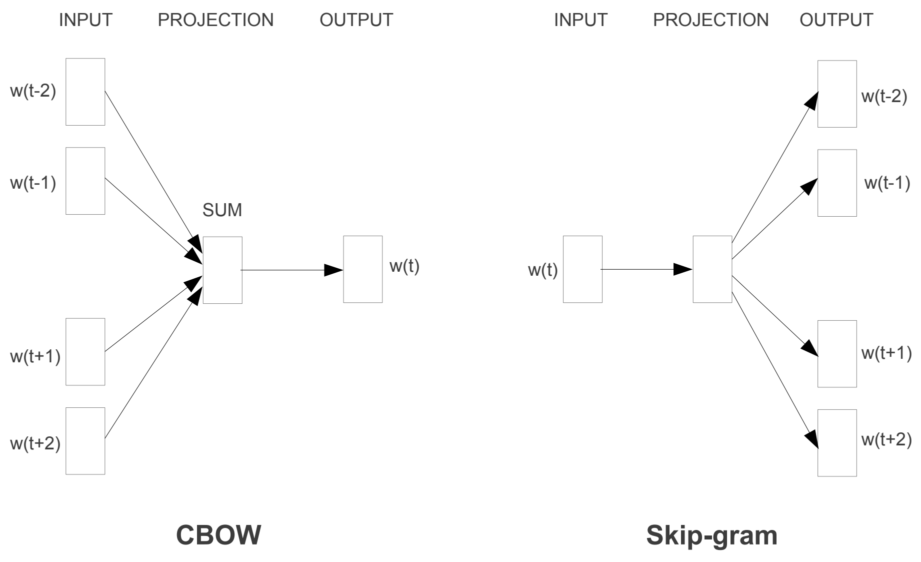

Figure 12: The CBOW and skip-gram models (Mikolov et al., 2013a)

technique proposed in Gutmann and Hyvärinen (2012). Negative sampling simply tries to maximize the probability assigned to the correct word to predict while minimizing the probability assigned to a number of randomly selected words each time. These random words are known as negative samples, which gives the name to the method. In any case, both hierarchical softmax and negative sampling make use of stochastic gradient descent and the backpropagation algorithm discussed in Section 3.4.3 for training. In addition to that, Mikolov et al. (2013c) also introduced a **subsampling for frequent words**, reducing the weight of words that occur very frequently in the training corpus (e.g. “the”, “of”, “in”), which are presumed to provide less information than rare words, by discarding words according to some probability that depends on their frequency and a parameter of the model.

The cosine similarity between embeddings is typically used to get a semantic similarity measure of the words they represent, although the Euclidean distance has also been used. The cosine similarity corresponds to the cosine of the angle between the given vectors v and w, and is typically computed taking their normalized dot product

$$\cos(v, w) = \frac{v \cdot w}{\|v\| \|w\|}$$

where $\|\cdot\|$ denotes the Euclidean norm. However, this is not the only interesting property that the word embeddings learned this way have, as performing simple arithmetic operations over them also gives surprisingly meaningful results. In particular, these embeddings have shown to be very effective for making analogies, both at the syntactic and semantic level. For example, vec(“biggest”) $-$ vec(“big”) $+$ vec(“small”) $\approx$ vec(“smallest”) (i.e. the resulting vector is very close to the word embedding of “smallest”, typically more close than to any other as measured by cosine similarity), and vec("France") − vec("Paris") + vec("Germany") ≈ vec("Berlin") (Mikolov et al., 2013a). Moreover, simple addition of vectors have also shown to work reasonably well. For instance, vec("Czech") + vec("currency") ≈ vec("koruna"), and vec("Vietnam") + vec("capital") ≈ vec("Hanoi") (Mikolov et al., 2013c). Even though these results might look quite surprising at first, some authors have given insight about why this word vector arithmetic works, suggesting further improvements for it (Levy and Goldberg, 2014a).

Finally, even though count models and predict models have been traditionally seen as two opposite approaches, it has been recently shown, both theoretically and empirically, that the optimization objective of these last neural predict models can be roughly expressed in terms of a traditional count model, so both approaches would mainly differ in the computational method to build the same underlying model. This way, Levy and Goldberg (2014b) show that the skip-gram model trained with negative sampling is equivalent to performing an implicit factorization of an occurrence matrix whose cells are the positive pointwise mutual information (PMI) of their corresponding word and context pairs, shifted by a global constant.

4.2 Word clustering

Word clustering consists in grouping words together in such a way that words that belong to the same group or class (known as cluster) are similar to each other in some syntactic and/or semantic sense. For instance, a good clustering would group words like "Germany", "Italy", "Russia" and "Ireland" into one class and words like "pig", "dog", "cow" and "horse" into another different class. These word clustering methods are mostly based on the distributional hypothesis, as they try to group words that are likely to occur in similar contexts.

This section discusses three of the most relevant word clustering methods, which have also been used in this project. Section 4.2.1 describes a classical clustering approach known as Brown clustering or IBM clustering. Section 4.2.2 presents the clustering method proposed in Clark (2003), which also incorporates morphological information. Finally, Section 4.2.3 discusses a simple approach for obtaining clusters from word embeddings.

4.2.1 Brown clustering

Brown clustering (Brown et al., 1992), also known as IBM clustering, is a hard agglomerative method for word clustering. It was originally proposed as part of a class-based n-gram model, where the probability of words is based on their respective clusters to address the sparsity problem of the classical n-gram models discussed in Section 2.3, although it can also be used for other tasks apart from language modeling. This way, the clustering is represented by a function π that assigns a class $c \in C$ to any given word $w \in V$, $|C| \ll |V|$, and a bigram language model is defined according to it:

$$P(w_1, \dots, w_m; \pi) = \prod_{i=1}^{m} P(w_i|\pi(w_i))P(\pi(w_i)|\pi(w_{i-1}))$$

where relative frequency estimates are used for the probabilities:

$$P(w_i|c_i) = \frac{\text{count}(w_i)}{\text{count}(c_i)}$$

$$P(c_i|c_{i-1}) = \frac{\text{count}(c_{i-1}, c_i)}{\text{count}(c_{i-1})}$$

Based on this, the quality of a clustering π is defined in terms of the probability it assigns to the training corpus. This meets the intuition of maximum likelihood estimation, as the best clustering according to this definition would also be the one that maximum likelihood estimation would choose. More concretely, the logarithm of the above probability is used normalized by the length of the text (Liang, 2005):

$$\begin{aligned}
\text{Quality}(\pi) &= \frac{1}{m} \log P(w_1, \dots, w_m; \pi) \\
&= \frac{1}{m} \sum_{i=1}^{m} \log \left[P(w_i|\pi(w_i))P(\pi(w_i)|\pi(w_{i-1}))\right] \\
&= \sum_{w,w'} \frac{\text{count}(w, w')}{m} \log \left(P(w'|\pi(w'))P(\pi(w')|\pi(w))\right) \\
&= \sum_{w,w'} \frac{\text{count}(w, w')}{m} \log \frac{\text{count}(w')}{\text{count}(\pi(w'))} \frac{\text{count}(\pi(w), \pi(w'))}{\text{count}(\pi(w))} \\
&= \sum_{w,w'} \frac{\text{count}(w, w')}{m} \log \frac{\text{count}(w')}{m} + \sum_{w,w'} \frac{\text{count}(w, w')}{m} \log \frac{m \,\text{count}(\pi(w), \pi(w'))}{\text{count}(\pi(w)) \,\text{count}(\pi(w'))} \\
&= \sum_{w'} \frac{\text{count}(w')}{m} \log \frac{\text{count}(w')}{m} + \sum_{c,c'} \frac{\text{count}(c, c')}{m} \log \frac{m \,\text{count}(c, c')}{\text{count}(c) \,\text{count}(c')}
\end{aligned}$$

Since, using frequency estimates as stated before, $P(w) = \frac{\text{count}(w)}{m}$, $P(c) = \frac{\text{count}(c)}{m}$ and $P(c, c') = \frac{\text{count}(c,c')}{m}$, this can be rewritten as follows:

$$\text{Quality}(\pi) = \sum_{w} P(w) \log P(w) + \sum_{c,c'} P(c, c') \log \frac{P(c, c')}{P(c)P(c')} = -H + I(\pi)$$

where $I(\pi)$ is the mutual information between adjacent clusters and H is the entropy of the word distribution.

This way, given that the entropy of the word distribution is fixed for the training corpus, the optimal clustering will be that maximizing the mutual information between adjacent

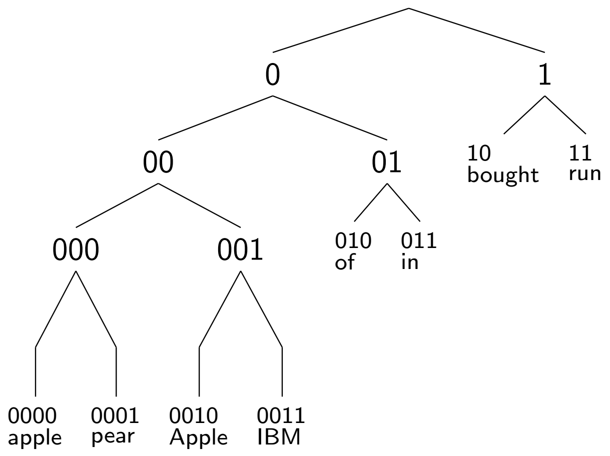

Figure 13: An example dendrogram produced by Brown clustering (Šuster, 2013)

clusters. Since it is computationally unfeasible to perform an exhaustive search, Brown et al. (1992) propose a greedy heuristic that starts with $|V|$ clusters (one per word) and iteratively merges them until obtaining the desired amount of clusters. More concretely, each iteration considers every possible cluster pair and merges the one for which the loss in the mutual information is the least. A naive implementation of this has a complexity of $O(|V|^5)$, which Brown et al. (1992) are able to reduce to $O(|V|^3)$ using some optimizations.

This greedy algorithm is a form of hierarchical clustering, so its output can also be seen as a binary tree where nodes correspond to the merges, commonly known as dendrogram. Thanks to this, the desired number of clusters can be obtained by cutting the tree at the appropriate level, without the need to rerun the algorithm. Moreover, each word is uniquely identified by its path in the tree, which can be expressed using a straightforward binary notation, and prefixes of these paths can also be used as features in different tasks. An example illustration of all this is given in Figure 13.

4.2.2 Clark clustering

Clark clustering (Clark, 2003) incorporates morphological evidence in addition to the distributional evidence into the clustering process. This is aimed to improve the clustering of words with very occurrences in the training corpus, in particular with morphologically rich languages in mind, and it was originally motivated for part of speech induction, although it can also be applied for other tasks.

The basic clustering model is very similar to that of Brown clustering in that it is also formalized as a class-based bigram language model and the algorithm tries to find the class assignment that maximizes the probability of the training corpus. However, instead of the greedy optimization method of Brown et al. (1992), Clark (2003) uses the exchange algorithm proposed by Martin et al. (1998), which starts with an initial clustering with the desired number of classes and iteratively improves it by moving each word from its current cluster to the one that gives the maximum increase in likelihood, if any.

Based on this basic model, Clark clustering introduces an additional prior probability for the partition π as follows:

$$P(\pi) = \prod_{c \in C} \prod_{\pi(w)=c} \alpha_c P_c(w)$$

In the above formulation, P_c is a character based model for each cluster c, which Clark clustering models through HMM, and $P_c(w)$ thus denotes the probability of the word w to belong to the class c according to its HMM model. The purpose of this is to introduce a bias that pushes morphologically similar words to the same cluster. For instance, the HMM model of a given class might assign high probabilities to words ending in "ly" or "ing", pushing those words that match these patterns, which are likely to be similar in morphological terms (in this case, adverbs derived from adjectives and present participles, respectively), into that particular cluster. In addition to that, α_c denotes the prior probability of each cluster c, which is estimated by dividing the number of different words in that cluster by the size of the vocabulary. This pushes rare words to clusters with a large number of different words, matching the intuition that a word with very few occurrences is more likely to belong to an open class like proper names than to a close class like prepositions.

4.2.3 Word embedding clustering

Given that, as discussed in Section 4.1, distances between word embeddings correlate with semantic similarity, any clustering algorithm can be applied over them to obtain meaningful word clusters. A straightforward approach, integrated in the word2vec package itself, is to use the well known k-means algorithm for that purpose. k-means (MacQueen et al., 1967) tries to find the partitioning π that minimizes the sum of the squared distances from each word to the centroid of its cluster, which is given by

$$\sum_{c \in C} \sum_{\pi(w)=c} \|w - \mu_c\|^2$$

where μ_c is the centroid of cluster c (i.e. the mean of the word vectors in that cluster).

Finding the optimal partitioning for a given number of clusters is an NP-hard problem, so heuristic methods are commonly used instead. A standard approach is to iteratively move all the words to the cluster whose centroid is closest to them, starting with some random points.

4.3 Bilingual word embeddings

Bilingual word embeddings are dense, low-dimensional representations of words in two languages in a common vector space. This way, if distances between monolingual word embeddings correlate with their semantic similarity, bilingual word embeddings attempt

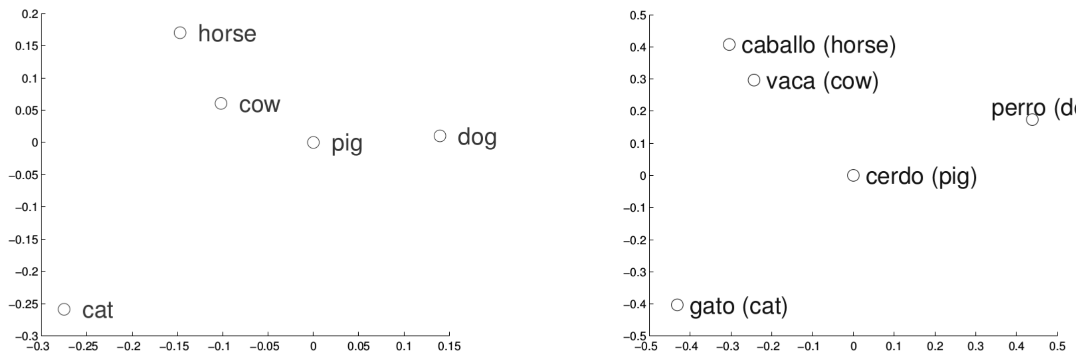

Figure 14: PCA visualization of two linearly related word spaces trained independently (Mikolov et al., 2013b)

to bring this to bilingual settings so that distances between embeddings of words in different languages also correlate with their semantic similarity.

There have been several proposals for this in the recent years, which can be broadly classified into three groups: **bilingual mapping**, which learns the embeddings of both languages independently and then finds a mapping between them, **monolingual adaptation**, which learns the embeddings of one language independently and then constrains the training of the other language, and **bilingual training**, which jointly learns the embeddings of both languages in a shared space (Luong et al., 2015). The following subsections analyze each approach one by one.

4.3.1 Bilingual mapping

Mikolov et al. (2013b) observe that independently trained word embeddings have a similar distribution for equivalent terms in two languages. An example of this can be seen in Figure 14, where PCA is used to visualize the same set of concepts in two languages, showing that they both have a similar geometrical arrangement. Based on this, they hypothesize that there exists a linear mapping between both spaces, and try to find such optimal mapping from a small bilingual dictionary. More concretely, given a set of word pairs and their corresponding embeddings $\{x_i, z_i\}$, they try to find the transformation matrix W so that Wx_i approximates z_i, minimizing the sum of squared distances between them:

$$\arg\min_W \sum_i \|Wx_i - z_i\|^2$$

For that purpose, they use the gradient descent algorithm presented in Section 3.3.

However, Xing et al. (2015) argue that, while this optimization objective minimizes the Euclidean distance, it is the cosine similarity that is commonly used, as it is the case of Mikolov et al. (2013b) themselves in their evaluation. Based on this, they propose an alternative approach that first constraints the training of the monolingual word embeddings to normalize them (i.e. they force the Euclidean norm of the word vectors to be 1) and then try to find the transformation matrix W that maximizes the sum of the inner products between the embedding pairs in the dictionary. When the word vectors are normalized, this effectively maximizes the average cosine similarity between these embedding pairs. However, even if the training forces the source vector x_i and the target vector z_i to be unit vectors, it would in principle be possible that Wx_i is not. In order to overcome this issue, they constraint W to be an orthogonal matrix, which preserves the inner product and therefore the Euclidean norm, giving place to the following optimization objective:

$$\arg\max_{W} \sum_{i} (Wx_i)^T z_i \qquad \text{s.t.} \quad W^T W = I$$

This is a constrained optimization problem and, rather than computing the exact solution, Xing et al. (2015) look for a simple approximation using a modified version of gradient descent. More concretely, after computing the gradient and updating the weights at each iteration, they set W to the orthogonal matrix that is closest it, which can be obtained by taking the Singular Value Decomposition (SVD) of it and replacing the singular values with ones.

Finally, while the above methods learn a single transformation from one language to the other, Faruqui and Dyer (2014) use one linear mapping for each language to project them to a common space. More concretely, they choose the projection that maximizes the correlation between the embedding pairs in the bilingual dictionary using canonical correlation analysis (CCA).

4.3.2 Monolingual adaptation

While the methods in the previous section find a mapping between two independently trained word vector spaces, an alternative approach is to train the language with the most resources on its own and then constrain the training of the other language to keep the bilingual correspondence. The rationale behind that is that not only would the word embeddings learned this way be in the same vector space, but the language with the most resources would also guide the training of the less resourced one, obtaining potentially better word embeddings for it.

Zou et al. (2013) propose one such approach based on a word aligned bilingual corpus (see Section 2.2). They first train word embeddings in English over a large monolingual corpus, and initialize the Chinese ones combining them according to the translation probabilities from the word alignment. They then train the Chinese word embeddings in a larger monolingual corpus starting from this initialization and incorporating an additional term in the optimization objective that accounts for the translation equivalences given by the word alignment.

4.3.3 Bilingual training

As seen in the previous sections, there are several methods that learn word embeddings in both or one of the languages independently and then bring them to a common space either by using a linear transformation or constraining the training of the other language. However, most approaches to build bilingual word embeddings do not do it in separate steps and try to jointly learn the embeddings of both languages instead, incorporating the appropriate constraints to ensure the bilingual correspondence.

Just as Zou et al. (2013) do for monolingual adaptation, many methods in this group make use of word aligned corpora. For instance, Klementiev et al. (2012) use the neural language model proposed in Bengio et al. (2003) to learn word embeddings in each language, incorporating an additional regularization term through multitask learning (Cavallanti et al., 2010) to push word pairs with a high translation probability according to word alignment to be close to each other. Another interesting approach is the BiSkip model proposed in Luong et al. (2015), which extends the skip-gram model discussed in Section 4.1 to learn bilingual word embeddings. For that purpose, in addition to training the model to predict the context of each word in its language, they also predict the context of the word that is aligned with it in the other language.

An alternative approach proposed in Kočiskỳ et al. (2014) is to jointly learn the word embeddings in both languages and the alignment itself. For that purpose, they use an adaptation of the FastAlign alignment model (Dyer et al., 2013), which is itself based on IBM Model 2 (see Section 2.2), where word translation probabilities are directly computed over their embeddings. Just as with IBM Models, training is done through expectation maximization (EM), fixing the alignments to get better translation probabilities (and, consequently, word embeddings) and fixing the translation probabilities to get better word alignments at each iteration.

Some other methods are also based on parallel corpora but do not use any word alignment at all. Lauly et al. (2014) propose an autoencoder based approach that is trained to reconstruct the bag-of-words representation of a given sentence and its corresponding translation in the parallel corpus. Overcoming one of the main limitations of the approaches discussed so far, the BilBOWA model proposed in Gouws et al. (2014) is able to train on large monolingual data and only needs a smaller parallel corpus. It is based on the skip-gram model trained with negative sampling, to which it adds a sampled bag-of-words bilingual objective as an additional regularization term.

Finally, there are some other methods that do not need any parallel corpora and use a small bilingual dictionary instead. The BARISTA model proposed in Gouws and Søgaard (2015) mix the source and the target monolingual corpora, randomly replace words in both languages according to the bilingual dictionary, and train a CBOW model on it. In addition to dictionaries with actual translation equivalences like "house" - "casa" (house), they also experiment with categorical equivalences like "car" - "casa" (both of which are nouns), learning bilingual word embeddings that are suitable for specific tasks like part-of-speech tagging or supersense tagging depending on the nature of these equivalences. Similarly, Wick et al. (2015) also use the CBOW model, to which they add a constraint term to encourage the embeddings in the bilingual dictionary to be close to each other, over the concatenation of both monolingual corpora, randomly replacing words in both languages according to the bilingual dictionary through a method they call artificial code-switching (ACS). This way, they argue that the CBOW model moves words in each language closer to their context, the bilingual constraint moves equivalent words in both languages closer to each other and ACS moves words in each language closer to the context of their equivalent word in the other language.

4.4 Distributional semantics for machine translation

Some of the distributional semantic techniques discussed so far have a direct application in machine translation. For instance, both Brown clustering and Clark clustering are formulated as class-based n-gram models, so they can be naturally used for language modeling in SMT (see Section 2.3). Similarly, some predict word embedding models are based on neural language models, so they can also be used for language modeling.

However, distributional semantics have also been applied in more subtle ways to overcome some of the limitations of other SMT components. One approach that has attracted a lot of attention in the last years, discussed in Section 4.4.1, is to add a new weight into the phrase-table that measures the semantic similarity of each phrase pair. Going one step further, distributional semantics have also been used to induce new translations of words or phrases that were not explicitly present in the parallel training data, which we discuss in Section 4.4.2. Other less-explored uses of continuous word and phrase representation in SMT include the reordering model (Li et al., 2013) or non-terminals in hierarchical systems (Wang et al., 2015).

4.4.1 Phrase translation similarity scoring

As discussed in Section 2.3.1, the phrase table of a standard log-linear phrase-based SMT system typically includes four feature functions: the forward translation probability, the inverse translation probability, the forward lexical weighting and the inverse lexical weighting. Recently, different methods have been proposed to add an additional score that, in one way or another, measures the semantic similarity of each phrase pair. This new weight would in principle be complementary to the standard translation probabilities and lexical weightings, which are both based on simple co-occurrence statistics, and the SMT system could therefore benefit from it.

Zou et al. (2013) propose a simple yet effective method for that based on bilingual word embeddings. First of all, they learn these bilingual word embeddings using their monolingual adaptation method presented in Section 4.3.2. They then compute the vector representation of phrases by simply taking the centroid of the embeddings of the words they contain, and add the cosine similarity between these centroids as a new weight into the phrase table. Using this technique, they get an improvement of nearly 0.5 BLEU points in the NIST08 Chinese-English translation task.

A notable limitation of the previous system is that the training of the word embeddings is independent from the task, composition method and similarity measure used (the centroid cosine similarity). Other methods try to overcome this issue by tightly integrating all these aspects into their training procedure. This way, Gao et al. (2014) use a feedforward neural network with two hidden layers that takes the bag-of-words representation of a phrase as input and produces a low-dimensional vector representation of the entire phrase as output. The same ANN is used for both languages so that their phrase embeddings are in the same vector space, and is trained to maximize the dot product between corresponding phrase pairs, which is then added into the phrase table as an additional weight. Using this technique, they get an improvement of up to 1.3 BLEU points in the WMT 2012 French-English translation task.

However, the previous method has another important limitation since, by taking the bag-of-words representation of phrases, it ignores the order of the words in them. For that reason, other architectures have been proposed to better model the compositionality of phrases. Among them, the Recursive Autoencoder (RAE) developed by Socher et al. (2011) has been successfully applied in SMT for different tasks that go beyond phrase translation similarity scoring, including reordering (Li et al., 2013) and non-terminal representation in hierarchical SMT (Wang et al., 2015). Proposed originally for sentiment analysis, a RAE consists of an encoding layer that combines two individual embeddings into a single one and a decoding layer that reconstructs the original embeddings from the latter. This way, starting with the word embeddings of a phrase, a greedy algorithm is used to iteratively combine contiguous embeddings minimizing their reconstruction error, obtaining a single embedding that represents the entire phrase at the end. In their original work, Socher et al. (2011) add a softmax layer on top of it for predicting the sentiment label distribution of the phrase in question, and the word embeddings, encoder, decoder and prediction layers are jointly trained to minimize both the prediction and regularization errors.

Zhang et al. (2014) adapt this model for phrase translation similarity scoring. For that purpose, they make use of two RAEs, one for each language, and adapt their prediction layer to project the phrase embeddings built with them to the other language. For each entry in the phrase table, they then compute the semantic distance in each direction as the squared Euclidean distance between the projected phrase embedding in one language and the phrase embedding in the other, and add these two weights to the phrase table. In order to train the model, they obtain positive examples from the training corpus by applying forced decoding and negative examples by taking random words as translations, and define a max-margin loss function over them. They test their method in Chinese-English translation, obtaining an average improvement of about one BLEU point over different test sets and up to 1.7 points in the best case. In a later work, Su et al. (2015) extend this model to better enforce structural alignment consistency according to word alignment, obtaining an improvement of about 0.7 BLEU points over the basic model in Zhang et al. (2014).

Finally, there have been other proposals that try to exploit the same underlying idea using other ANN architectures. For instance, Cho et al. (2014) develop a RNN encoder-decoder that consists of two recurrent neural networks (RNN). The encoder RNN encodes a variable length source phrase into a fixed length vector representation in a way similar to a RAE, but it does so sequentially instead of following a tree-like structure. However, rather than encoding both phrases and computing the distance between them, Cho et al. (2014) use a decoder RNN that generates the target language phrase from the encoded source language phrase, and use it to compute the probability of the target phrase given the source phrase, which they add into the phrase table. At the same time, they use a novel activation function for both RNNs that adaptively remembers and forgets past history in a way similar to a long short-term memory (LSTM), but in a much simpler way. Using this method, they get an improvement of 0.57 BLEU points in the WMT 2014 English-French translation task, which goes up to 1.34 BLEU points when used together with a neural language model, showing that both techniques are complementary.

4.4.2 New translation induction

As seen in Section 4.3.1, bilingual mapping methods for word embeddings learn some transformation that projects the source language embeddings into the target language space. Since the word embeddings in both languages are trained independently, this transformation can also be applied to words that are missing or have few occurrences in the bilingual training data as long as they do appear in the monolingual corpus, and it can therefore be used to induce new translations that were not explicitly seen before.

In their original work, Mikolov et al. (2013b) themselves show the potential of this idea with the translation induction experiment they do. More concretely, they train English, Spanish and Czech word embeddings using the CBOW architecture and monolingual data from WMT 2011. After that, they generate bilingual dictionaries with the 5,000 most frequent words in them and their translation as given by Google Translate, and use each of these bilingual dictionaries to learn a transformation matrix between different embedding spaces with their method. For the subsequent 1,000 most frequent words, they project their corresponding embeddings into the target language space using the previously learned transformation matrix, and take the closest word embeddings there as measured by cosine similarity as their translation. They then compare these induced translations with those given by Google Translate, obtaining a precision of over 30% for both English-Spanish directions and over 20% for both English-Czech directions. Moreover, when measuring the top 5 accuracy instead of the top 1, these percentages increase to over 50% for English-Spanish and over 40% for English-Czech.

Subsequent works propose improvements for this basic approach and apply them in end-to-end machine translation. This way, Zhao et al. (2015) use several local linear projections instead of a single global one as Mikolov et al. (2013b) do, and propose the use of redundant bit vectors to speed up the retrieval of the nearest neighbors in the target language space. Moreover, they extend their method from words to phrases by simply taking the element-wise addition of word embeddings. They then apply this method to generate new translation rules for an Arabic-English and an Urdu-English phrase-based SMT system, obtaining an improvement of up to 1.6 and 0.5 BLEU points, respectively.

4.5 Conclusions

In this chapter, we have discussed word embeddings and word clusters, two distributional semantic techniques that represent words by dense vectors and discrete classes, respectively. These representations are useful to mitigate the sparsity problem in natural language processing while providing a means to incorporate distributional knowledge acquired from large monolingual corpora. For that reason, in Chapter 6 we integrate both word embedding and word cluster features in the logistic regression model we propose in Chapter 5 for dynamic phrase translation probability scoring. Apart from that, we have seen different approaches to learn bilingual extensions of word embeddings and their application in SMT through phrase translation similarity scoring and new translation induction. Along these lines, in Chapter 7 we propose a new framework to learn bilingual word embedding mappings and test them on a word translation induction task, and in Chapter 8 we explore the direct use of these and other bilingual word embeddings for phrase translation similarity scoring through different metrics.

5 Logistic regression for dynamic phrase translation probability modeling

In this chapter, we propose the use of logistic regression for dynamic phrase translation probability modeling. Our model allows to flexibly define a set of features to characterize phrase translation candidates, and the probability that the logistic regression classifier predicts for them is dynamically added into the phrase table. We show that, for the right set of features, this probability is equivalent to the relative frequency counting estimate used in phrase-based SMT and, therefore, our model can be seen as a generalization of the standard translation probabilities that allows to naturally incorporate additional features.

This chapter presents the proposed model together with a basic set of lexical features, and tests them experimentally in English-Spanish translation. Later in Chapter 6, we explore the incorporation of distributional semantic features into the model through word clusters and word embeddings.

This way, we describe our model in Section 5.1, and prove its equivalence with relative frequency counting in Section 5.2. Section 5.3 discusses our basic feature design, which comprises source language lexical features, target language lexical features and their combination. Section then 5.4 presents the experiments on English-Spanish translation and discusses the obtained results. Finally, Section 5.5 concludes the chapter.

The work in this chapter was done in collaboration with researchers in the Institute of Formal and Applied Linguistics (ÚFAL) at Charles University in Prague, especially with Aleš Tamchyna, to whom we would like to thank their support and help. More concretely, our implementation is based on their Moses extension to integrate Vowpal Wabbit, which is now part of the official Moses release. At the same time, both teams shared our experimental settings, results and findings. In any case, the formalization and proof of relative frequency counting equivalence, the implementation of some variants and features, the experimental framework used and all the experiments presented here are completely original. At the same time, our work to integrate distributional semantic features presented in Chapter 6 is also completely original.

5.1 Proposed model

Let f_i be a binary feature function that, given a source language phrase $\bar{f}$ and a target language phrase $\bar{e}$, yields 1 if the phrase pair $(\bar{f}, \bar{e})$ has a given property and 0 otherwise. For instance, we could define a feature function that tells whether $\bar{e}$ contains the bigram "la casa", or another one that tells if $\bar{e}$ contains the word "casa" and $\bar{f}$ contains the word "house".

For a given set of features, we define a logistic regression model to estimate the probability of a given phrase $\bar{f}$ being translated as $\bar{e}$ as follows (see Section 3.3):

$$h(\bar{f}, \bar{e}) = \frac{1}{1 + e^{-\sum_i w_i f_i(\bar{f}, \bar{e})}}$$

This predicted probability is then added into the log-linear feature combination of a standard phrase-based SMT system along with the forward and inverse translation probabilities and lexical weightings (see Section 2.3.1), either statically or, in our case, dynamically, which allows to use context dependent features.

However, it might be argued that this integration model is flawed in that it does not guarantee a true probability distribution over the possible translations of a given phrase. For that reason, we also consider the variant where the softmax function is used to add a normalized forward probability into the phrase table:

$$h_{\text{softmax}}(\bar{f}, \bar{e}) = \frac{e^{\sum_i w_i f_i(\bar{f}, \bar{e})}}{\sum_{\bar{e}'} e^{\sum_i w_i f_i(\bar{f}, \bar{e}')}}$$

The original implementation by ÚFAL researchers, which is the basis of our work, used this latter integration method. Nevertheless, we think that this model has its own drawbacks, similar to those of the standard translation probabilities. In particular, if a phrase has very few entries in the phrase table, this model will tend to predict higher probabilities for each of them, whereas if a phrase has many different entries, it will tend to predict lower probabilities. In the most extreme case, for a phrase with only one possible translation in the phrase table, the model will necessarily predict a probability of 1, no matter how adequate it is. In other words, it assumes that one, and only one, of the entries in the phrase table is the correct translation for a given phrase and context, but it is possible that, in reality, none or several of them are. As a consequence, this integration method will tend to overestimate the probability of long phrases, which will necessarily have less occurrences in the training corpus than their corresponding subphrases and are therefore more likely to miss an appropriate translation for a given context. For that reason, we also implement the direct integration of the logistic regression probability, which we prefer for our theoretical discussion here, and test them both later in the experiments in Section 5.4.

In order to train the model, for every occurrence of a phrase $\bar{f}$ in the training corpus that is aligned with $\bar{e}$, we create a positive example $(\bar{f}, \bar{e})$ and all the possible negative examples $(\bar{f}, \bar{e}')$ where $\bar{e}' \neq \bar{e}$ and there is an entry for $(\bar{f}, \bar{e}')$ in the phrase table. The original implementation by ÚFAL researchers does not consider the case where $\bar{f}$ is unaligned or its translation $\bar{e}$ is missing in the phrase table. However, we consider that, for the direct integration method discussed above, creating negative examples for these cases could help to better model the probability of phrases that are not often translated into the target language (e.g. due to ellipsis), and test both variants in our experiments in Section 5.4. In either case, the model is trained to find the set of weights $\hat{w}$ that minimizes the error function of logistic regression (see Section 3.3):

$$\begin{aligned}
\hat{w} &= \arg\min_{w} -\sum_{(\bar{f},\bar{e})} \left(y(\bar{f},\bar{e}) \log h(\bar{f},\bar{e}) + \left(1 - y(\bar{f},\bar{e})\right) \log\left(1 - h\left(x^{(i)}\right)\right)\right) \\
&= \arg\min_{w} \sum_{(\bar{f},\bar{e})} \left(y(\bar{f},\bar{e}) \log \frac{1}{h(\bar{f},\bar{e})} + \left(1 - y(\bar{f},\bar{e})\right) \log \frac{1}{1 - h\left(x^{(i)}\right)}\right) \\
&= \arg\min_{w} \sum_{(\bar{f},\bar{e})} \left(y(\bar{f},\bar{e}) \log\left(1 + e^{-\sum_i w_i f_i(\bar{f},\bar{e})}\right) + \left(1 - y(\bar{f},\bar{e})\right) \log \frac{1}{1 - \frac{1}{1+e^{-\sum_i w_i f_i(\bar{f},\bar{e})}}}\right) \\
&= \arg\min_{w} \sum_{(\bar{f},\bar{e})} \left(y(\bar{f},\bar{e}) \log\left(1 + e^{-\sum_i w_i f_i(\bar{f},\bar{e})}\right) + \left(1 - y(\bar{f},\bar{e})\right) \log \frac{1 + e^{-\sum_i w_i f_i(\bar{f},\bar{e})}}{e^{-\sum_i w_i f_i(\bar{f},\bar{e})}}\right) \\
&= \arg\min_{w} \sum_{(\bar{f},\bar{e})} \left(y(\bar{f},\bar{e}) \log\left(1 + e^{-\sum_i w_i f_i(\bar{f},\bar{e})}\right) + \left(1 - y(\bar{f},\bar{e})\right) \log\left(1 + e^{\sum_i w_i f_i(\bar{f},\bar{e})}\right)\right)
\end{aligned}$$

where $y(\bar{f},\bar{e})$ is 0 for negative examples and 1 for positive ones.

Since this corresponds to the maximum likelihood estimation for the given observations, it can be shown that, for a feature set composed solely of unique phrase pair identifiers, the model will learn the translation probabilities $\phi(\bar{e}|\bar{f})$ estimated by relative frequency that are used in standard phrase-based SMT (see Section 2.3). In other words, if we define a feature function f_i for every phrase pair $(\bar{f},\bar{e})$ so that $f_i(\bar{f},\bar{e}) = 1$, $\forall j \neq i, f_j(\bar{f},\bar{e}) = 0$, and $\forall (\bar{f}',\bar{e}') \neq (\bar{f},\bar{e}), f_i(\bar{f}',\bar{e}') = 0$, then the following will hold[3] (see Section 5.2 for a step by step proof):

$$h(\bar{f},\bar{e}) = \phi(\bar{e}|\bar{f}) = \frac{\text{count}(\bar{f},\bar{e})}{\text{count}(\bar{f})}$$

Therefore, the proposed model can be seen as a generalization of the standard translation probabilities used in SMT that allows to naturally incorporate additional features to obtain better probability estimates. We hypothesize that this could be useful in several ways:

1. It allows to use additional lexical features shared across different phrase pairs that could have a smoothing effect and help to better estimate the probability of phrase pairs with very few occurrences in the training corpus. For instance, let's say that the phrase pair "Real Sociedad won the match - la Real ganó el partido" has very few occurrences in the training corpus, making it hard to estimate its probability, whereas "Real Sociedad lost the match - la Real perdió el partido" and "Athletic won the match - el Athletic ganó el partido" occur more often. If we had a feature

[3]Note that, in Section 2.3, we defined $\phi(\bar{f},\bar{e}) = \frac{\text{count}(\bar{f},\bar{e})}{\sum_{\bar{f}} \text{count}(\bar{f},\bar{e})}$ instead. This is because the standard translation probabilities do not consider unaligned phrases, whereas we explore both the variant that does and does not. Therefore, in our notation here count$(\bar{f})$ denotes the total number of considered occurrences, including unaligned ones or not depending on the variant used.

for every combination of bigrams between the source and the target languages, we could potentially get better probability estimates thanks to the additional evidence we have for bigram pairs like “Real Sociedad - la Real”, “won the - ganó el” or “the match - el partido”. What is more, this would even make it possible to estimate the probability of new phrase pairs with no occurrences in the training corpus. We design such lexical features later in Section 5.3 and test them in the experiments in Section 5.4.

2. It allows to incorporate features for the context of the phrase in the source language, overcoming one of the most obvious limitations of traditional translation models. Sometimes it is the context that determines the appropriateness of a phrase pair as it is the case of “your arm - tu arma” and “your arm - tu brazo”. Traditional translation models would nonetheless assign a fixed score to each pair and nothing but the language model could choose the correct translation taking the context into account. The proposed model can overcome this issue by replacing the static weights of traditional translation models with dynamic ones that depend on context features. We also design such lexical context features in Section 5.3 and test them in the experiments in Section 5.4. It should be noted that, while it is theoretically possible to have context features for both the source and the target languages, we do not explore the latter option in this work due to the extra complexity that it would pose for decoding, as the context would not be known a priori (see Section 5.3.2).

3. It provides a framework to naturally incorporate linguistic information into the translation model. Factored models are typically used for this purpose, combining independent translation and generation models for different factors such as the surface form, the lemma and the part-of-speech (see Section 2.4.1). By adding the relevant features, the proposed model allows to use the very same information with the added advantage that the different factors can be more flexibly combined and be tuned according to a joint optimization objective. Based on this idea, we explore the incorporation of distributional semantic features into the model through word clusters and word embeddings later in Chapter 6.

5.2 Proof of relative frequency counting equivalence

Following the formulation in Section 5.1, we define a feature set composed of unique phrase pair identifier feature functions f_i so that, for every f_i associated to a phrase pair $(\bar{f}, \bar{e})$, $f_i(\bar{f}, \bar{e}) = 1$, $\forall j \neq i, f_j(\bar{f}, \bar{e}) = 0$, and $\forall (\bar{f}', \bar{e}') \neq (\bar{f}, \bar{e}), f_i(\bar{f}', \bar{e}') = 0$. Since $\forall (\bar{f}', \bar{e}') \neq (\bar{f}, \bar{e}), f_i(\bar{f}', \bar{e}') = 0$, the weight w_i associated with the feature function f_i will be annulled for every $(\bar{f}', \bar{e}') \neq (\bar{f}, \bar{e})$. At the same time, given that $\forall j \neq i, f_j(\bar{f}, \bar{e}) = 0$, both the error function and the prediction of the phrase pair $(\bar{f}, \bar{e})$ will only depend on the weight w_i. As a consequence, each weight w_i can be optimized independently, and it is enough to consider the phrase pair $(\bar{f}, \bar{e})$ characterized by its corresponding feature function f_i for that.

Since, considering the training method described above, we will have $\text{count}(\bar{f}, \bar{e})$ positive examples for the phrase pair $(\bar{f}, \bar{e})$ (one for each occurrence of $\bar{f}$ in the training corpus that is aligned with $\bar{e}$) and $\text{count}(\bar{f}) - \text{count}(\bar{f}, \bar{e})$ negative examples (one for each occurrence of $\bar{f}$ in the training corpus that is not aligned with $\bar{e}$), the optimal weight $\hat{w}_i$ will be given by

$$\begin{aligned}
\hat{w}_i &= \arg\min_{w_i} \left(\text{count}(\bar{f}, \bar{e}) \log\left(1 + e^{-w_i}\right) + \left(\text{count}(\bar{f}) - \text{count}(\bar{f}, \bar{e})\right) \log\left(1 + e^{w_i}\right)\right) \\
&= \arg\min_{w_i} \left(\text{count}(\bar{f}, \bar{e}) \log\left(\frac{1 + e^{w_i}}{e^{w_i}}\right) + \left(\text{count}(\bar{f}) - \text{count}(\bar{f}, \bar{e})\right) \log\left(1 + e^{w_i}\right)\right) \\
&= \arg\min_{w_i} \left(\text{count}(\bar{f}, \bar{e}) \left(\log\left(1 + e^{w_i}\right) - w_i\right) + \left(\text{count}(\bar{f}) - \text{count}(\bar{f}, \bar{e})\right) \log\left(1 + e^{w_i}\right)\right) \\
&= \arg\min_{w_i} \left(\text{count}(\bar{f}) \log\left(1 + e^{w_i}\right) - \text{count}(\bar{f}, \bar{e}) w_i\right)
\end{aligned}$$

In order to find the weight w_i that minimizes the above expression analytically, we will first calculate its derivative:

$$\frac{d}{dw_i} \left(\text{count}(\bar{f}) \log\left(1 + e^{w_i}\right) - \text{count}(\bar{f}, \bar{e}) w_i\right) = \text{count}(\bar{f}) \frac{e^{w_i}}{1 + e^{w_i}} - \text{count}(\bar{f}, \bar{e})$$

Since the error function is continuous and differentiable, it is known that its global minimum will either happen when the derivative is 0 or as w_i approaches $-\infty$ or ∞. Let's first calculate the values of w_i for which the former holds:

$$\begin{aligned}
\text{count}(\bar{f}) \frac{e^{w_i}}{1 + e^{w_i}} - \text{count}(\bar{f}, \bar{e}) &= 0 \\
\text{count}(\bar{f}) e^{w_i} &= \text{count}(\bar{f}, \bar{e}) \left(1 + e^{w_i}\right) \\
e^{w_i} &= \frac{\text{count}(\bar{f}, \bar{e})}{\text{count}(\bar{f}) - \text{count}(\bar{f}, \bar{e})} \\
w_i &= \log \frac{\text{count}(\bar{f}, \bar{e})}{\text{count}(\bar{f}) - \text{count}(\bar{f}, \bar{e})}
\end{aligned}$$

In order to know what happens as w_i approaches -∞, we will calculate the corresponding limit:

$$\begin{aligned}
&\lim_{w_i \to -\infty} \left(\text{count}(\bar{f}, \bar{e}) \log\left(1 + e^{-w_i}\right) + \left(\text{count}(\bar{f}) - \text{count}(\bar{f}, \bar{e})\right) \log\left(1 + e^{w_i}\right)\right) \\
&= \lim_{w_i \to -\infty} \text{count}(\bar{f}, \bar{e}) \log\left(1 + e^{-w_i}\right) = \begin{cases} \infty & \text{count}(\bar{f}, \bar{e}) > 0 \\ 0 & \text{count}(\bar{f}, \bar{e}) = 0 \\ -\infty & \text{count}(\bar{f}, \bar{e}) < 0 \end{cases}
\end{aligned}$$

Similarly, we will calculate the limit of the error function as w_i approaches ∞:

$$\lim_{w_i \to \infty} \big(\text{count}(\bar{f}, \bar{e}) \log \big(1 + e^{-w_i}\big) + \big(\text{count}(\bar{f}) - \text{count}(\bar{f}, \bar{e})\big) \log \left(1 + e^{w_i}\right)\big)$$

$$= \lim_{w_i \to \infty} \big(\text{count}(\bar{f}) - \text{count}(\bar{f}, \bar{e})\big) \log \left(1 + e^{w_i}\right) = \begin{cases} \infty & \text{count}(\bar{f}) > \text{count}(\bar{f}, \bar{e}) \\ 0 & \text{count}(\bar{f}) = \text{count}(\bar{f}, \bar{e}) \\ -\infty & \text{count}(\bar{f}) < \text{count}(\bar{f}, \bar{e}) \end{cases}$$

This way, when $\text{count}(\bar{f}) > \text{count}(\bar{f}, \bar{e}) > 0$ the error function will go to ∞ as w_i approaches $-\infty$ and ∞, so $w_i = \log \frac{\text{count}(\bar{f},\bar{e})}{\text{count}(\bar{f}) - \text{count}(\bar{f},\bar{e})}$ will necessarily be the global minimum. The cases where $\text{count}(\bar{f}, \bar{e}) < 0$ or $\text{count}(\bar{f}) < \text{count}(\bar{f}, \bar{e})$ can be ignored, as it is not possible to have a negative count by definition. For the missing cases, it can be seen that, when $\text{count}(\bar{f}, \bar{e}) = 0$, the expression $\log \frac{\text{count}(\bar{f},\bar{e})}{\text{count}(\bar{f}) - \text{count}(\bar{f},\bar{e})}$ gives $-\infty$, so it also corresponds to the global minimum of the error function. Finally, when $\text{count}(\bar{f}) = \text{count}(\bar{f}, \bar{e})$, the expression $\log \frac{\text{count}(\bar{f},\bar{e})}{\text{count}(\bar{f}) - \text{count}(\bar{f},\bar{e})}$ gives ∞ and thus corresponds to the global minimum of the error function as well. Therefore, we have shown that, for every possible case, $w_i = \log \frac{\text{count}(\bar{f},\bar{e})}{\text{count}(\bar{f}) - \text{count}(\bar{f},\bar{e})}$ is the value that minimizes the loss function.

Now that we know the optimal value of the weight w_i, we can replace it in the sigmoid function to obtain the probability predicted by the model:

$$h(\bar{f}, \bar{e}) = \frac{1}{1 + e^{-w_i}} = \frac{1}{1 + e^{-\log \frac{\text{count}(\bar{f},\bar{e})}{\text{count}(\bar{f}) - \text{count}(\bar{f},\bar{e})}}} = \frac{1}{1 + \frac{\text{count}(\bar{f}) - \text{count}(\bar{f},\bar{e})}{\text{count}(\bar{f},\bar{e})}} = \frac{\text{count}(\bar{f}, \bar{e})}{\text{count}(\bar{f})}$$

The last expression corresponds to the relative frequency estimate of the translation probability $\phi(\bar{e}|\bar{f})$ used in standard phrase-based systems, so it is therefore proved that, for this set of features, $h(\bar{f}, \bar{e}) = \phi(\bar{e}|\bar{f})$.

5.3 Feature design

In this section, we describe the lexical features we design for the proposed model. We define different source and target language features independently in Section 5.3.1 and 5.3.2, respectively, and analyze the need and approach to combine them in Section 5.3.3.

5.3.1 Source language features

Our lexical feature set for the source language consists of the following independent features:

- **Indicator**: A unique identifier for the source phrase $\bar{f}$ so that its corresponding feature function f_i is 1 for that phrase and 0 for the rest.
- **Phrase internal**: A position independent identifier for each word in the source phrase that is shared among all the source phrases containing the word in question.

	Mary [did not slap] the green witch
Indicator	`sind^did_not_slap:1`
Phrase internal	`sin^did:1, sin^not:1, sin^slap:1`
Window (size=2)	`c^-1^Mary:1, c^1^the:1, c^2^green:1`
Window (size=2, bow=1)	`c^bow^Mary:1, c^bow^the:1, c^bow^green:1`
Bag of words	`bow^Mary:1, bow^did:1, bow^not:1, bow^slap:1, bow^the:1, bow^green:1, bow^witch:1`

Table 1: Example source language lexical features

In other words, the phrase internal feature function f_i associated with a given word will be 1 for every source phrase that contains that word and 0 for the rest.

- **Window**: A position dependent identifier for each word in the context window of the source phrase, so that the window feature function f_i associated with a given word and offset (e.g. the word "from" occurring two positions before the phrase) is 1 for all the instances (regardless of the specific source phrase) that have it in the corresponding position of their contexts and 0 for the rest. The size of the window is parametrized. In addition to that, we also implement a variant without the offset information, using a bag of words representation of the context window instead.

- **Bag of words**: A position independent identifier for each word in the segment (typically sentence) in which the phrase occurs, including the phrase itself. In other words, the bag of words feature function associated with a given word will be 1 for all the segments that contain that word and 0 for the rest.

As it can be seen, the first two characterize the phrase itself, whereas the last two are given by the context in which it occurs. Table 1 shows the features that would be accordingly generated for an example source phrase and context.

5.3.2 Target language features

In principle, the target language lexical features could be defined analogously to the source language ones described in Section 5.3.1. Even if this would be indeed possible for training, the use of context features would nonetheless pose a problem for testing, where the context of a phrase translation candidate is not known a priori. More concretely, this would make the score and, by extension, the choice of a target phrase depend on the choice of its surrounding target phrases, which would at the same time depend on the choice of the former and other target phrases. However, as discussed in Section 2.3, SMT decoding is an NP-complete problem (Knight, 1999), so considering all the target phrase candidate combinations is intractable in practice. It would therefore be necessary to integrate dynamic phrase translation scores in the heuristic search that is used instead, which would not only be very challenging to implement, but could even have a negative effect due to the extra complexity that it would pose. For that reason, we only define context independent features for the target phrase as follows:

	Maria [no daba una bofetada] a la bruja verde
Indicator	`tind^no_daba_una_bofetada:1`
Phrase internal	`tin^no:1, tin^daba:1, tin^una:1, tin^bofetada:1`

Table 2: Example target language lexical features

- **Indicator**: A unique identifier for the target phrase $\bar{e}$ so that its corresponding feature function f_i is 1 for that phrase and 0 for the rest.
- **Phrase internal**: A position independent identifier for each word in the target phrase that is shared among all the target phrases containing the word in question. In other words, the phrase internal feature function f_i associated with a given word will be 1 for every target phrase that contains that word and 0 for the rest.

Table 2 shows the features that would be accordingly generated for an example target phrase, aligned with the one shown in Table 1 for the source side.

5.3.3 Feature combination

Even if the source and target language features defined throughout the section serve to characterize a phrase pair $(\bar{f}, \bar{e})$, a logistic regression classifier trained over them would not be able to perform well due to the linear separability problem discussed in Section 3.4. The reason is that, by definition, the output of a linear model like logistic regression is a function of a linear combination of the features it takes, which implies that the contribution of each feature to the final prediction is independent to the rest. As a consequence, the classifier would need to assess the adequateness of the source phrase and its corresponding target phrase independently, without considering the relation between them, so it would not be able to model their translation probability at all, as we aim. For instance, in the example above "did not slap" and "no daba una bofetada" or "the green witch" and "la bruja verde" would be scored independently, so logistic regression would not be able to assign a high probability to the correct phrase pairs ("did not slap", "no daba una bofetada") and ("the green witch", "la bruja verde") while assigning a low probability to the incorrect phrase pairs ("did not slap", "la bruja verde") and ("the green witch", "no daba una bofetada").

In order to overcome this issue, we combine the source and target language features to create new interaction features that capture the relation between both sides, following the feature engineering approach discussed in Section 3.4.1. More concretely, we take the Cartesian product between the source and the target feature sets, creating a new interaction feature for every possible combination between a source language feature $f_i(\bar{f})$ and a target language feature $f_j(\bar{e})$ as $f_{i,j}(\bar{f}, \bar{e}) = f_i(\bar{f}) \times f_j(\bar{e})$.

5.4 Experiment and results

In order to implement the proposed logistic regression model, we use the Vowpal Wabbit large scale machine learning system presented in Section 3.6, and integrate it in the Moses

	Use	Domain	Sentences	English tokens	Spanish tokens
Europarl	training	parliamentary proceedings	1,929,366	51,593,389	54,021,555
WMT12	development	news	3,000	72,895	78,831
WMT13	test	news	3,000	64,761	70,485

Table 3: Bilingual English-Spanish corpora

	Domain	Sentences	Tokens
Europarl	parliamentary proceedings	1,965,800	57,053,435
United Nations	administrative documents	11,197,000	361,653,831
News Commentary	news	174,500	5,118,381
News 2007-2012	news	13,384,600	385,990,821
Common Crawl	crawling	1,845,300	49,308,828
KDE4	software localizations	218,700	2,350,928
OpenOffice	software localizations	38,100	481,341

Table 4: Monolingual Spanish corpora used for language modeling

toolkit presented in Section 2.6 as a dynamic feature function. As discussed before, this implementation is based on that of ÚFAL researchers, which is now part of the official Moses release, with the necessary changes for the variants and new features that we propose.

We test our method in English-Spanish translation using a standard experimental setup. We first train a baseline phrase-based SMT system using Moses, and then measure the contribution of the logistic regression model over it under different settings. Our baseline system uses standard parameters: ixa-pipe-tok for tokenization (Agerri et al., 2014), MGIZA for word alignment with the grow-diag-final-and symmetrization heuristic, a maximum length of 80 tokens per sentence and 5 tokens per phrase, translation probabilities in both directions with Good Turing discounting, lexical weightings in both directions, a phrase length penalty, a lexicalized reordering model and a target language model. Table 3 summarizes the details of the parallel corpora used for that purpose. As for the language model, we trained a 5-gram model for each monolingual corpus in Table 4, and interpolate them by optimizing perplexity on the WMT12 development set. The weights for the different components were adjusted to optimize BLEU using MERT tuning over the WMT12 development set, with an n-best list of size 100. However, given the stochastic nature of MERT, this introduces some variability in the results. In order to reduce it, we perform 3 independent runs for each configuration. We use BLEU as our evaluation metric (see Section 2.5), and report the average score over the 3 MERT runs. Table 5 shows the results we obtain for the baseline system under these conditions.

In order to train the logistic regression model we propose with Vowpal Wabbit, we extract all the possible positive and negative examples from the training set as described in Section 5.1. We use its cost sensitive one-against-all classification mode with label dependent features and namespace-dependent square features to efficiently implement the feature set we propose, using the hashing trick for feature vectorization with a 28 bit table to avoid collisions. We run the training for 30 epochs using stochastic gradient descent with a decay learning rate and feature caching, running a local spanning tree server to parallelize the process. In order to prevent overfitting, the model resulting after each epoch is tested

	BLEU (%)	
	Development	**Test**
Baseline	31.68	28.28

Table 5: Results on English-Spanish translation for the baseline system

in a separate validation set, choosing the one with the highest accuracy on it. We try using both the development corpus and a holdout set of 5,000 sentences that we keep aside from training for that purpose. While using the development corpus has the advantage of being in the same domain as the test set and consistent with the choice for MERT tuning, it also has the disadvantage of containing many phrases that were not seen during training and are therefore missing in the phrase table, which is the reason why we try to experiment with both options.

Under these conditions, running each experiment takes 1-2 days in an entire 8 core node of our cluster, a Xeon E5 2640 v2 with 128GB of RAM. Since the cluster has only 4 such general purpose nodes to be shared among the entire research group, this posed a very important limitation on the number of experiments we could carry out. For that reason, we were unable to perform a systematic exploration for the different parameters, and had to follow an opportunistic approach instead.

We decided to use the indicator, phrase internal and position dependent window of size 3 as the source language features based on the experimental results reported by ÚFAL researchers for Czech-English. We also ran some preliminary experiments with larger window sizes, but did not appreciate any improvement. As for the target side, using the indicator feature combined with the source language features is roughly equivalent to having an independent classifier for each target phrase, whereas using the phrase internal feature combined with the source language features is roughly equivalent to having an independent classifier for each target word. For that reason, we tried using both features on their own as well as their combination. The results obtained for these experiments are given in Table 6. They were all done using the direct integration of the logistic classifier probabilities, considering unaligned or missing translations for negative examples, and using the development set to choose the best epoch. The results show a small improvement over the baseline for all the 3 settings both in the development and the test set, with the combination of the target indicator and phrase internal achieving the best performance. This suggests that both target features add useful and complementary information for phrase translation probability modeling, so we use them both for all the other experiments that we perform.

In addition to that, we also analyzed, as discussed before, the effect of using a random holdout from the training corpus, as opposed to the development corpus, for the validation set to choose the best epoch. The results for this experiment are given in Table 7, showing slightly better performance for the training holdout configuration.

Finally, we also tested the different integration methods discussed in Section 5.1 when it comes to the use of the softmax function for probability normalization as opposed to the direct used of the logistic regression prediction, and the use or not of unknown translations

TARGET FEATURES		**INTEGRATION**			**BLEU (%)**	
Indicator	**Phrase internal**	**Softmax**	**Unknown examples**	**Validation**	**Development**	**Test**
	X		X	Development	31.84 (+0.16)	28.39 (+0.11)
X			X	Development	31.76 (+0.08)	28.38 (+0.10)
X	X		X	Development	**31.88 (+0.20)**	**28.44 (+0.16)**

Table 6: Results on English-Spanish translation for different target language features

TARGET FEATURES		**INTEGRATION**			**BLEU (%)**	
Indicator	**Phrase internal**	**Softmax**	**Unknown examples**	**Validation**	**Development**	**Test**
X	X		X	Development	31.88 (+0.20)	28.44 (+0.16)
X	X		X	Train holdout	**31.94 (+0.26)**	**28.53 (+0.25)**

Table 7: Results on English-Spanish translation for different validation sets

(i.e. unaligned source phrases or target phrases missing in the phrase table) to generate negative examples. Table 8 shows the results for this experiment. As it can be seen, both variations we propose (i.e. the direct integration of the logistic regression probability and the use of unknown examples) perform better than the original model from ÚFAL researchers, with the improvement being particularly notable in the development set.

All in all, our best configuration achieves an improvement of 0.26 and 0.25 BLEU points over the baseline in the development and the test set, respectively. While it cannot be said that this improvement is big in quantitative terms, it is quite consistent across the different settings tested, getting an improvement of over 0.16 BLEU points in the test set for all the configurations that use both target features. As a reference, it is said that it is necessary to have an improvement of at least 0.5 BLEU points for a machine translation work to be accepted in a major conference like ACL. While we still need to make some progress to get there, it is remarkable that, in contrast with many works in the field, our improvement does not come from incorporating external linguistic information into the translation process, but relies solely on the bilingual corpora used to train the SMT system itself, so the method can be directly applied to any other language pair regardless of the available resources.

In order to better understand where this improvement actually comes from, Figure 15 shows the weights that MERT assigns to the different components for the baseline system and our best performing configuration. As it can be seen, our logistic regression model almost completely replaces the forward translation probability, and it also reduces

TARGET FEATURES		**INTEGRATION**			**BLEU (%)**	
Indicator	**Phrase internal**	**Softmax**	**Unknown examples**	**Validation**	**Development**	**Test**
X	X			Train holdout	31.84 (+0.16)	28.46 (+0.18)
X	X		X	Train holdout	**31.94 (+0.26)**	**28.53 (+0.25)**
X	X	X		Train holdout	31.75 (+0.07)	28.48 (+0.20)

Table 8: Results on English-Spanish translation for different integration methods

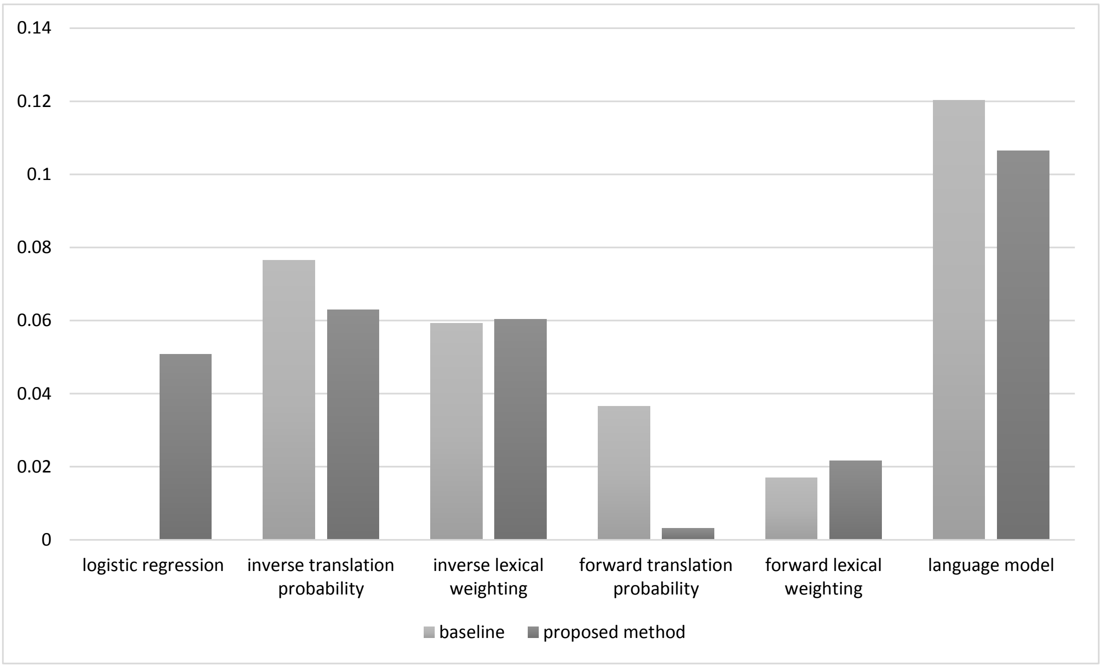

Figure 15: Average weights assigned by MERT for the different components

the weight of the inverse translation probability and the language model. This suggests that the probability estimates given by our model are better than the standard translation probabilities used in phrase-based SMT. As discussed in Section 5.1 and proved in Section 5.2, the proposed model can be seen as a generalization of these frequency counting based translation probabilities that allows to incorporate additional features to increase its expressive power, so these results are consistent with our theoretical expectations. At the same time, we only use source language context features and generate negative examples for the different phrase table entries for each source phrase in the training corpus, so it makes sense that the logistic regression model replaces the forward translation probability more than the inverse one, as this is what it formally models. We also attempted to adapt the training procedure to model the inverse translation probability instead, but our preliminary experiments did not show any positive result. Finally, the fact that our model considers the source language context to score each phrase pair explains why it also complements the language model for lexical selection.

5.5 Conclusions

In this chapter, we have proposed the use of logistic regression for dynamic phrase translation probability modeling. We have proved that our model is a generalization of the relative frequency counting used in phrase-based SMT, allowing to naturally incorporate additional features to obtain better probability estimates. We define source and target language lexical features independently and take their quadratic combination, including features shared across different phrase pairs to mitigate the sparsity problem as well as source language context features to mitigate the locality problem in SMT.

Our experiments on English-Spanish machine translation show the effectiveness of the proposed model, with an improvement of up to 0.25 BLEU points over a strong baseline when using this basic set of lexical features. While it cannot be said that this improvement is big in quantitative terms, it is remarkably consistent across the different settings tested and relies solely on the bilingual corpus used to train the baseline SMT system itself. Our analysis of the weights assigned by MERT reveals that this improvement comes from almost completely replacing the forward translation probabilities while partly taking the place of the inverse translation probabilities and the language model, suggesting that the probability estimates given by our model are indeed better than those used in standard phrase-based SMT and pointing to the usefulness of context features for lexical selection.

We conclude that the proposed method provides a promising framework to incorporate additional information into the translation model, as we do next in Chapter 6 with distributional semantics, serving as a much more flexible alternative to factored models that is even capable of dealing with dynamic context-dependent features.

6 Distributional semantic features for phrase translation logistic regression

In this chapter, we incorporate distributional semantic features into the logistic regression model presented in Chapter 5. As discussed there, the proposed model provides a flexible framework to naturally incorporate linguistic information into the translation model, and this is what we explore in this chapter for distributional semantics through word clusters and word embeddings. This way, we present the word cluster and word embedding features we propose for that purpose in Section 6.1 and 6.2, respectively, discuss the experiments performed with them on English-Spanish translation and the obtained results in Section 6.3, and outline our conclusions in Section 6.4.

6.1 Word cluster features

As discussed in Section 4.2, word clustering groups words together in such a way that those that belong to the same cluster are similar to each other in some syntactic and/or semantic sense. These clusters are usually learned in an unsupervised manner from large monolingual corpora, based on the distributional hypothesis.

Given that word clusters essentially map each word into one of the predefined number of classes, we define word cluster features by replacing the surface forms in all the features proposed in Section 5.3 by their corresponding class identifier. Table 9 shows an example of this for the source language features. Since we learn word clusters in a separate monolingual corpus, we use a special class for out-of-vocabulary (OOV) words, as it is the case of "Mary" in this artificial example.

It should be noted that these word cluster features are not able to increase the expressive power of the model in the training set itself. The reason is that the mapping defined by word clustering is static, so these new features and their corresponding weights act as shared weights for the previously presented lexical features. As a consequence, when it comes to the training corpus, for any model using a feature set of both lexical and word cluster features, there exists a completely equivalent one that does not use the word cluster features at all.

However, this does not mean that the word cluster features cannot be helpful. In fact, shared weights are characteristic of convolutional neural networks, where they are used to aid the generalization capacity of the model by enforcing the property of translation invariance. We think that, in our case, shared weights could also be helpful for guiding the training and learn models that generalize better. In particular, we hypothesize that word cluster features could be useful in the following aspects:

- **They allow to incorporate unsupervised knowledge from large monolingual corpora.** Traditional phrase-based SMT systems use a target monolingual corpus for language modeling, but are unable to benefit from source monolingual corpora. Thanks to word cluster features, we are able to incorporate knowledge acquired in an

	Mary|cloov [**did**|cl53 **not**|cl17 **slap**|cl87] **the**|cl23 **green**|cl49 **witch**|cl83
Indicator	`sind^cl53_cl17_cl87:1`
Phrase internal	`sin^cl53:1, sin^cl17:1, sin^cl87:1`
Window (size=2)	`c^-1^cloov:1, c^1^cl23:1, c^2^cl49:1`
Window (size=2, bow=1)	`c^bow^cloov:1, c^bow^cl23:1, c^bow^cl49:1`
Bag of words	`bow^cloov:1, bow^cl53:1, bow^cl17:1, bow^cl87:1, bow^cl23:1, bow^cl49:1, bow^cl83:1`

Table 9: Example source language word cluster features

unsupervised manner from monolingual corpora in either language into the model. This knowledge can be useful to guide the training process, in particular when it comes to words with very few occurrences in the bilingual corpus, for which the additional evidence from monolingual corpora can be very helpful. Related to that, the notion of shared weights underlying word cluster features can push the training to find more general regularities, reducing the risk of overfitting.

- **They allow to account for out-of-vocabulary (OOV) words.** For the lexical features proposed in Section 5.3, words that do not appear in the training corpus do not have any associated weight, so they are simply ignored by the model. Nevertheless, thanks to word cluster features every word that is found in the monolingual corpus can have an associated cluster with its corresponding weight, even if it does not appear in the bilingual corpus. Moreover, even for words that do not occur in the monolingual corpus, the special OOV class mentioned before can be used.

- **They allow to drastically reduce the number of features for some information, aiding generalization.** The sparsity of natural language results in a massive number of different lexical features. As a consequence, adding some relevant information into the model might actually have a negative impact due to overfitting, as is likely to occur with the position dependent window feature above certain limit for the window size. In contrast, word cluster features provide a natural way to incorporate the same information while reducing the number of different features to the desired amount, learning models that generalize better.

6.2 Word embedding features

As discussed in Section 4.1, word embeddings are dense, low-dimensional representations of words in a continuous vector space distributed in some meaningful way from the semantic point of view. Therefore, just as word clusters provide a static mapping from words to discrete classes, word embeddings provide a static mapping from words to continuous vectors. Even if we have so far only dealt with binary features, continuous features like specific dimension of these word embeddings can also be incorporated into the proposed model by simply multiplying their value by their corresponding weight. The motivation for this is similar to that of word cluster features, as word embeddings also provide a way to incorporate knowledge acquired from large monolingual corpora. Moreover, they have

	Mary [did not slap] the green witch
Phrase concatenation	`vec^in^1^0:0.084, vec^in^2^0:-0.045, vec^in^3^0:0.227, ...`
Phrase centroid	`vec^in^c^0:0.089, ...`
Window concatenation (size=2)	`vec^b^1^0:-0.079, vec^a^1^0:0.147, vec^a^2^0:0.037, ...`
Window centroid (size=2)	`vec^b^c^0:-0.079, vec^a^c^0:0.092, ...`

Table 10: Example source language word embedding features. For brevity, only the first dimension is shown.

the added advantage that, unlike word clusters, they provide a unique representation for each word, while overcoming the sparsity problem by using a dense, low-dimensional vector space. We define the following features that are based on them:

- **Phrase concatenation**: The concatenation of the embeddings of all the words in the phrase in the order in which they occur. A global numeric feature is used for each dimension of the resulting embedding. Equivalently, this can be seen as a position dependent embedding representation of each word in the phrase.

- **Phrase centroid**: The average or centroid embedding of all the words in the phrase. A global numeric feature is used for each dimension of the resulting embedding.

- **Window concatenation**: The concatenation of the embeddings in each side of the context window of the phrase from the closest to the farthest. The size of the window is parametrized, and a global numeric feature is used for each dimension of the resulting embedding in each side. Equivalently, this can be seen as a position dependent embedding representation of each word in the context window of the phrase.

- **Window centroid**: The average or centroid embedding for all the words in each side of the context window of the phrase. The size of the window is parametrized, and a global numeric feature is used for each dimension of the resulting embedding in each side.

As it can be seen, the first two characterize the phrase itself and we therefore define them for both the source and the target side independently, whereas the last two are given by the context in which the phrase occurs, so we only use them for the source language following the discussion in Section 5.3.2. Table 10 shows the source side embedding features that would be accordingly generated for an example phrase and context.

6.3 Experiment and results

In order to test the distributional semantic features presented throughout the section, we use the same experimental settings presented in Section 5.4. Since the new features do not aim to replace the basic lexical features used there, but rather complement them, we combine them both under different configurations and measure the impact in the BLEU score. Due to the high computational cost already discussed there (each experiment needs

CLUSTERS			INTEGRATION		BLEU (%)	
Corpus	**Method**	**Size**	**Softmax**	**Unknown examples**	**Development**	**Test**
gigaword	word2vec	200	X		31.76 (+0.08)	28.43 (+0.15)
reuters	clark	600	X		**31.83 (+0.15)**	28.39 (+0.11)
reuters	brown	1000	X		31.82 (+0.14)	**28.46 (+0.18)**

Table 11: Results on English-Spanish translation for different word clusters

a minimum of 1-2 days taking an entire 8 core node of our cluster), we were not able to perform a systematic exploration of the different feature and parameters either and had to follow an opportunistic approach instead. Moreover, we also experimented with the different variants proposed, partly for the purpose of contrasting the results obtained at that section when different features are used, and partly as a consequence of the order in which the experiments were actually carried out.

As far as the word cluster features are concerned, we reused different word clusterings from a yet to be published work in named entity recognition in collaboration with Rodrigo Agerri, to whom we would like to thank all his help. Based on the results in that task, we chose one configuration in the source language (English) for each of the three clustering techniques presented in Section 4.2. In the case of Brown clustering, we used 1000 word classes learned from the Reuters RCV1 corpus, consisting of around 63 million words that were reduced to 35 after preprocessing. The same corpus was used for Clark clustering, but in that case the selected number of clusters was 600. Finally, the word embedding clustering was done with word2vec on the 5th edition of the Gigaword corpus, which consists of 4000 million words, and the number of classes was 200.

In order to test the effect of these clustering settings in our task, we used them for the source language indicator, phrase internal and the position dependent window feature of size 5, along with the basic lexical features used in the experiments in Section 5.4 (the source and target indicator and phrase internal, and the position dependent source window of size 3), combining them as described in Section 5.3.3. In all the cases, we used the original softmax integration method without unknown examples, taking the training holdout for the validation set to choose the best epoch.

The obtained results are given in Table 11. As it can be seen, the observed differences are small and not conclusive given the the contrast between the development set and the test set and the variability introduced by MERT. For that reason, we decide to use the configuration that performs best in the development set (Clark clustering in the Reuters RCV1 corpus with 600 word classes) for the rest of our experiments.

Using this clustering configuration, we next analyzed the effect of using different feature sets. Table 12 summarizes the different settings tested and their corresponding BLEU in the development and test sets. In all the cases, we used the lexical indicator and phrase internal features for the target language with the softmax integration method without unknown examples, taking the training holdout for the validation set. The results suggest that the word clustering features cannot replace the lexical features completely, since the second configuration, which does not use any lexical feature for the source language, obtains

SOURCE FEATURES (WORD / CLUSTER)				**INTEGRATION**		**BLEU (%)**	
Indicator	**Phrase internal**	**Window**	**Bag of words**	**Softmax**	**Unknown examples**	**Development**	**Test**
both	both	3/5	-	X		31.83 (+0.15)	28.39 (+0.11)
cluster	cluster	0/5	-	X		31.75 (+0.07)	28.30 (+0.02)
both	both	3/5	cluster	X		31.71 (+0.03)	**28.45 (+0.17)**
both	both	3/3	-	X		31.81 (+0.13)	28.38 (+0.10)
word	word	3/5	-	X		**31.85 (+0.17)**	28.44 (+0.16)

Table 12: Results on English-Spanish translation for different word cluster features

SOURCE FEATURES (WORD / CLUSTER)				**INTEGRATION**		**BLEU (%)**	
Indicator	**Phrase internal**	**Window**	**Bag of words**	**Softmax**	**Unknown examples**	**Development**	**Test**
both	both	3/5	-			31.86 (+0.18)	28.46 (+0.18)
both	both	3/5	-		X	**31.98 (+0.30)**	28.45 (+0.17)
both	both	3/5	-	X		31.83 (+0.15)	28.39 (+0.11)
word	word	3/5	-			31.77 (+0.09)	28.47 (+0.19)
word	word	3/5	-		X	31.92 (+0.24)	**28.55 (+0.27)**
word	word	3/5	-	X		31.85 (+0.17)	28.44 (+0.16)

Table 13: Results on English-Spanish translation for different word cluster integration methods

considerably worse results, in particular in the test set. For the remaining configurations the results are much less conclusive, but we observe that, in this case, using a window of 5 for the cluster features performs slightly better than using a window of 3, and using surface forms alone for the indicator and phrase internal features also gives better results than using both word clusters and surface forms. Finally, it is remarkable that the configuration using the bag of words feature for word clusters obtains the best results in the test set in spite of being the worst one in the development set.

In addition to that, we also tested the different integration methods as done in Section 5.4 for the two best performing feature sets above in the development set. The obtained results are shown in Table 13. Once again, we observe that the variants we propose perform better than the original model. This way, the direct integration of logistic regression probabilities with unknown examples obtains the best results so far both in the development set and the test set, in the first case using both surface forms and word clusters for the source language indicator and phrase internal features and in the second case using the word clusters alone for the same features.

As for the word embedding features, we faced huge difficulties to carry out our experiments because of the big memory requirements and slow training times caused by the feature explosion when combining source language and target language features. Running a single experiment in a complete 8 core node of our cluster would have taken several weeks with standard 300-dimension embeddings even when using the centroid phrase or window features alone, an unfeasible cost for us considering that our cluster has only 4 such nodes to be shared among the entire research group. For that reason, we ran a single experiment with the source language window centroid feature of size 5 using 50-dimension

	INTEGRATION		BLEU (%)	
	Softmax	**Unknown examples**	**Development**	**Test**
Window centroid (size=5)		X	32.02 (+0.34)	28.44 (+0.16)

Table 14: Results on English-Spanish translation with embedding features

word embeddings, which took 5-6 days. More concretely, we trained our embeddings using the skip-gram model with word2vec, and we set the context window to 5, the subsampling to 1e-4, the number of negative samples to 30 and the number of iterations to 10 (see Section 4.1). In addition to Europarl, we used the English portion of the News 2007-2012 monolingual corpus from WMT as our training corpus, which has 68,521,621 sentences and 1,612,954,315 tokens. Our vocabulary consisted of all the words with at least one occurrence in Europarl together with all the words in the monolingual corpus with at least 5 occurrences in total. In addition to the source language window centroid of size 5 for the word embeddings, we also used our basic lexical features (the source and target indicator and phrase internal, and the position dependent source window of size 3), combining them as described in Section 5.3.3.

The obtained results are shown in Table 14. As it can be seen, this configuration achieves the best results so far in the development set, with an absolute improvement of 0.08 BLEU points with respect to the equivalent configuration without the word embedding features in Section 5.4. However, in spite of the good results in the development set, we do not get any improvement in the test set.

6.4 Conclusions

In this chapter, we have explored the integration of distributional semantic features into the logistic regression model presented in Chapter 5, which serves to incorporate unsupervised knowledge from large monolingual corpora, mitigate the problem of out-of-vocabulary words, and reduce the number of parameters of the model, aiding generalization. For that purpose, we have proposed the use word cluster features by replacing the surface forms in lexical features by their corresponding class, as well as word embedding features for both the phrase and its context through concatenation and averaging.

Using these features, we are able to improve the best results in Chapter 5 with the basic lexical features, but this improvement is very modest and insufficient to draw any clear conclusion. However, we consider that our work was greatly conditioned by the high computational cost of running the experiments along with the strong hardware limitations that we had, and we plan to keep running more experiments for different feature sets and hyperparameters so as to obtain more conclusive results, in particular for the word embedding features.

7 A general framework for bilingual word embedding mappings

In this chapter, we propose a general framework to learn bilingual word embedding mappings. Our model starts with a basic optimization objective and allows for several variants that we prove to be equivalent to other meaningful optimization goals. Moreover, we show that several existing methods fit naturally in this framework, providing a more global view of the different bilingual mapping techniques and, in some cases, revealing flaws in their theoretical justification.

We describe the proposed method and all its variants in Section 7.1. Section 7.2 then analyzes its relation to other bilingual mapping methods proposed in the literature. After that, Section 7.3 experimentally tests the proposed framework in comparison to other methods in an English-Italian word translation induction task, and Section 7.4 concludes the chapter. Later in Chapter 8, we apply the proposed method to phrase translation similarity scoring and test it on phrase translation selection and end-to-end machine translation.

7.1 Proposed method

In this section, we describe the general framework we propose to learn bilingual word embedding mappings. Section 7.1.1 first presents the basic optimization objective we propose along with an exact method to solve it, and later sections propose several variants that, without altering it in any fundamental way, are shown to be equivalent to other meaningful and conceptually relevant optimization goals. This way, Section 7.1.2 introduces the orthogonality constraint, which serves to guaranty monolingual invariance. Section 7.1.3 then discusses length normalization while Section 7.1.4 discusses mean centering, which serve to maximize the average cosine similarity and covariance, respectively. Finally, Section 7.1.5 describes how the propose framework also serves to work with weighted and partial dictionaries.

7.1.1 Basic optimization objective

Let X and Z denote the word embedding matrices in two languages for a given bilingual dictionary so that their ith row X_{i*} and Z_{i*} are the word embeddings of the ith entry in the dictionary. Our goal is to find a linear transformation matrix W so that XW best approximates Z. For that purpose, the optimization objective we propose minimizes the sum (or, equivalently, the average) of squared Euclidean distances for the dictionary entries:

$$\arg\min_{W} \sum_{i} \|X_{i*}W - Z_{i*}\|^2$$

Alternatively, this is equivalent to minimizing the (squared) Frobenius norm of the entire residual matrix:

$$\arg\min_{W} \|XW - Z\|_F^2$$

Consequently, W will be the so called least-squares solution of the linear matrix equation $XW = Z$. This is a known problem in linear algebra and can be solved by taking the Moore-Penrose pseudoinverse $X^+ = \left(X^T X\right)^{-1} X^T$ of matrix X as follows:

$$W = X^+ Z = \left(X^T X\right)^{-1} X^T Z$$

A computationally efficient way to calculate the pseudoinverse X^+ is to take the Singular Value Decomposition (SVD) of X as $X = U\Sigma V^T$, where U and V are orthogonal matrices and Σ is a diagonal matrix whose entries, listed in descending order by convention, are known as singular values. For such factorization, it can be shown that $X^+ = V\Sigma^+ U^T$ holds, where the pseudoinverse Σ^+ is simply the transpose of the matrix formed by replacing the non-zero elements in Σ with their reciprocal.

7.1.2 Orthogonality constraint for monolingual invariance

An extension of the above introduced optimization objective is to constrain W to be an orthogonal matrix (i.e. a square matrix with orthonormal columns and rows, so $W^T W = WW^T = I$ holds) which yields to the following constrained optimization problem:

$$\arg\min_{W} \sum_i \|X_{i*} W - Z_{i*}\|^2 \qquad \text{s.t.} \quad W^T W = I$$

So as to better understand the motivation behind this constraint, it should be noted that an orthogonal transformation preserves the dot product of any two vectors X_{i*} and X_{j*}, which can be trivially derived from the definition of orthogonality itself:

$$(X_{i*}W) \cdot (X_{j*}W) = (X_{i*}W)\,(X_{j*}W)^T = X_{i*}WW^T X_{j*}^T = X_{i*} I X_{j*}^T = X_{i*} X_{j*}^T = X_{i*} \cdot X_{j*}$$

Based on this, it can be easily seen that an orthogonal transformation also preserves the Euclidean norm, Euclidean distance and cosine similarity in the original vector space:

$$\|X_{i*}W\| = \sqrt{(X_{i*}W) \cdot (X_{i*}W)} = \sqrt{X_{i*} \cdot X_{i*}} = \|X_{i*}\|$$

$$\|X_{i*}W - X_{j*}W\| = \|(X_{i*} - X_{j*})W\| = \|X_{i*} - X_{j*}\|$$

$$\cos\left(X_{i*}W, X_{j*}W\right) = \frac{(X_{i*}W) \cdot (X_{j*}W)}{\|X_{i*}W\|\|X_{j*}W\|} = \frac{X_{i*} \cdot X_{j*}}{\|X_{i*}\|\|X_{j*}\|} = \cos\left(X_{i*}, X_{j*}\right)$$

Therefore, all the fundamental properties and relations in the original vector space are kept after an orthogonal transformation. Thanks to this, the orthogonality constraint guarantees that the monolingual word embeddings in the source language do not degrade after the mapping, producing the optimal bilingual mapping that is monolingually invariant. When doing so, it is also enforcing a relevant property that such transformation should intuitively have, which could be useful to avoid degenerated solutions and learn even better bilingual mappings.

However, this also turns the original unconstrained optimization problem into a constrained one, so the Moore-Penrose pseudoinverse method introduced above cannot be used anymore to solve it. Nevertheless, we next develop a simple yet efficient exact method that accounts for the orthogonality constraint. First of all, we transform the original minimization problem into a simpler and more convenient maximization one as follows:

$$\begin{aligned}
\arg\min_W \sum_i \|X_{i*}W - Z_{i*}\|^2 &= \arg\min_W \sum_i (X_{i*}W - Z_{i*}) \cdot (X_{i*}W - Z_{i*}) \\
&= \arg\min_W \sum_i \|X_{i*}W\|^2 + \|Z_{i*}\|^2 - 2\,(X_{i*}W) \cdot Z_{i*} \\
&= \arg\min_W \sum_i \|X_{i*}\|^2 + \|Z_{i*}\|^2 - 2X_{i*}WZ_{i*}^T \\
&= \arg\min_W \; -\sum_i X_{i*}WZ_{i*}^T \\
&= \arg\max_W \sum_i X_{i*}WZ_{i*}^T \\
&= \arg\max_W \; \mathrm{Tr}\left(XWZ^T\right) \\
&= \arg\max_W \; \mathrm{Tr}\left(Z^TXW\right)
\end{aligned}$$

In the above expression, $\mathrm{Tr}(\cdot)$ denotes the trace operator (the sum of all the elements in the main diagonal), and the last equality is given by its cyclic property. At this point, we can take the SVD of Z^TX as $Z^TX = U\Sigma V^T$, which yields to the following expression:

$$\arg\max_W \; \mathrm{Tr}\left(Z^TXW\right) = \arg\max_W \; \mathrm{Tr}\left(U\Sigma V^TW\right) = \arg\max_W \; \mathrm{Tr}\left(\Sigma V^TWU\right)$$

Since V^T, W and U are orthogonal matrices, their product V^TWU will also be an orthogonal matrix. In addition to that, given that Σ is a diagonal matrix, its trace after an orthogonal transformation will be maximal when the values in its main diagonal are preserved after the mapping, that is, when the orthogonal transformation matrix is the identity matrix. This will happen when $V^TWU = I$ in our case, which yields to the following solution for W:

$$\begin{aligned}
V^TWU &= I \\
VV^TWUU^T &= VIU^T \\
W &= VU^T
\end{aligned}$$

To sum up, the solution of the proposed optimization problem with the orthogonality constraint is given by $W = VU^T$, where $Z^TX = U\Sigma V^T$ is the SVD factorization of Z^TX. It is interesting to see that this is somehow similar, at least in algorithmic terms, to the solution of the unconstrained version, which recall that could be computed as $W = V\Sigma^+U^TZ$, where $X = U\Sigma V^T$ was the SVD factorization of X.

7.1.3 Length normalization for maximum cosine similarity

Another possible variation for the proposed optimization problem is to normalize all the word embeddings in both languages to be unit vectors, which can be done by simply dividing them by their Euclidean norm. We next show that, when W is constrained to be an orthogonal matrix, this effectively maximizes the sum of the cosine similarity between the dictionary entries:

$$\begin{aligned}
\arg\min_W \sum_i \left\| \frac{X_{i*}}{\|X_{i*}\|} W - \frac{Z_{i*}}{\|Z_{i*}\|} \right\|^2 &= \arg\min_W \sum_i \left(\frac{X_{i*}W}{\|X_{i*}\|} - \frac{Z_{i*}}{\|Z_{i*}\|} \right) \cdot \left(\frac{X_{i*}W}{\|X_{i*}\|} - \frac{Z_{i*}}{\|Z_{i*}\|} \right) \\
&= \arg\min_W \sum_i \frac{\|X_{i*}W\|^2}{\|X_{i*}\|^2} + \frac{\|Z_{i*}\|^2}{\|Z_{i*}\|^2} - \frac{2\left(X_{i*}W\right) \cdot Z_{i*}}{\|X_{i*}\|\|Z_{i*}\|} \\
&= \arg\min_W \sum_i \frac{\|X_{i*}\|^2}{\|X_{i*}\|^2} + \frac{\|Z_{i*}\|^2}{\|Z_{i*}\|^2} - \frac{2\left(X_{i*}W\right) \cdot Z_{i*}}{\|X_{i*}W\|\|Z_{i*}\|} \\
&= \arg\min_W \sum_i 2 - 2\cos\left(X_{i*}W, Z_{i*}\right) \\
&= \arg\max_W \sum_i \cos\left(X_{i*}W, Z_{i*}\right)
\end{aligned}$$

Therefore, the proposed method serves not only to find the orthogonal transformation minimizing the average squared Euclidean distance between the dictionary entries, but also the one maximizing their average cosine similarity, for which it is enough to normalize all the word embeddings in a preprocessing step. The motivation for this is twofold. First, length normalization brings all the training instances to the same scale, ensuring that they will contribute equally to the optimization goal unless some explicit weighting like the one we introduce in Section 7.1.5 is used. Second, given that cosine similarity is typically used to measure the similarity between different embeddings more than the Euclidean distance, optimizing for it is likely to be more consistent with respect to the intended application of the resulting embeddings.

7.1.4 Mean centering for maximum covariance

Analogously to length normalization, one can also mean center each word embedding in the target language, that is, make their mean be zero, which can be done by simply subtracting their actual mean. We next show that, when W is constrained to be an orthogonal matrix, such mean centering maximizes the sum (or average) covariance between the dictionary entries:

$$\begin{aligned}\arg\min_W \sum_i \|X_{i*}W - Z_{i*}C_n\|^2 &= \arg\min_W \sum_i \|X_{i*}W\|^2 + \|Z_{i*}C_n\|^2 - 2\,(X_{i*}W)\cdot(Z_{i*}C_n)\\ &= \arg\max_W \sum_i (X_{i*}W)\cdot(Z_{i*}C_n)\\ &= \arg\max_W \sum_i X_{i*}WC_n^T Z_{i*}^T\\ &= \arg\max_W \sum_i X_{i*}WC_nC_n^T Z_{i*}^T\\ &= \arg\max_W \sum_i (X_{i*}WC_n)\cdot(Z_{i*}C_n)\\ &= \arg\max_W \sum_i n\,\mathrm{cov}\,(X_{i*}W, Z_{i*})\\ &= \arg\max_W \sum_i \mathrm{cov}\,(X_{i*}W, Z_{i*})\end{aligned}$$

where C_n denotes the centering matrix, which mean centers any vector that is multiplied by, defined as $C_n = I_n - \frac{1}{n}J_n$, where n is the dimension of the word embeddings, I_n is the $n \times n$ identity matrix and J_n is the $n \times n$ all-ones matrix. The perhaps non-obvious equivalence $C_nC_n^T = C_n^T$ used above is given by the symmetry and idempotence properties of C_n, which can be easily derived based on this definition:

$$C_n^T = \left(I_n - \frac{1}{n}J_n\right)^T = I_n^T - \frac{1}{n}J_n^T = I_n - \frac{1}{n}J_n = C_n$$

$$\begin{aligned}C_n^2 = \left(I_n - \frac{1}{n}J_n\right)^2 &= I_n^2 + \left(\frac{1}{n}J_n\right)^2 - I_n\left(\frac{1}{n}J_n\right) - \left(\frac{1}{n}J_n\right)I_n\\ &= I_n + \frac{1}{n^2}J_n^2 - \frac{2}{n}J_n = I_n + \frac{1}{n^2}(nJ_n) - \frac{2}{n}J_n = I_n - \frac{1}{n}J_n = C_n\end{aligned}$$

and thus $C_nC_n^T = C_nC_n = C_n = C_n^T$.

A very similar reasoning can also be used to prove that, by mean centering all the word embeddings in the target language dimension-wise, it is the sum (or average) dimension-wise covariance that is maximized. This dimension-wise mean centering ensures that the expected product of two random embeddings in any dimension is zero. Consequently, this also makes the expected dot product between two random embeddings, which is equivalent to the cosine product if they are length normalized as discussed in Section 7.1.3, to be zero, capturing the intuition that two random words would not in principle be similar nor dissimilar between them.

In order to prove the stated equivalence, it is convenient to express the above minimization objective in terms of the Frobenius norm of the entire residual matrix:

$$\arg\max_W \sum_i \mathrm{cov}\,(X_{i*}W, Z_{i*}) = \arg\min_W \sum_i \|X_{i*}W - Z_{i*}C_n\|^2 = \arg\min_W \|XW - ZC_n\|_F^2$$

and observe that, just as ZC_n mean centers the rows (i.e. each word embedding) in Z, C_vZ alternatively mean centers its columns (i.e. each dimension of the word embeddings). Based on this, we can conveniently express the new optimization objective and prove the stated equivalence:

$$\begin{aligned}
\arg\min_W \|XW - C_vZ\|_F^2 &= \arg\min_W \sum_i \|XW_{*i} - C_vZ_{*i}\|^2 \\
&= \arg\min_W \sum_i \|XW_{*i}\|^2 + \|C_vZ_{*i}\|^2 - 2\left(XW_{*i}\right)\cdot\left(C_vZ_{*i}\right) \\
&= \arg\max_W \sum_i \left(XW_{*i}\right)^T\left(C_vZ_{*i}\right) \\
&= \arg\max_W \sum_i W_{*i}^T X^T C_v Z_{*i} \\
&= \arg\max_W \sum_i W_{*i}^T X^T C_v^T C_v Z_{*i} \\
&= \arg\max_W \sum_i \left(C_vXW_{*i}\right)\cdot\left(C_vZ_{*i}\right) \\
&= \arg\max_W \sum_i \operatorname{cov}\left(XW_{*i}, Z_{*i}\right)
\end{aligned}$$

where $\sum_i \|XW_{*i}\| = \|XW\|_F^2 = \|X\|_F^2$ holds thanks to the orthogonality constraint on W.

At the same time, it can be easily shown that mean centering the embeddings in both languages dimension-wise instead of the target language ones alone does not affect this optimization objective:

$$\begin{aligned}
\arg\min_W \|C_vXW - C_vZ\|_F^2 &= \arg\max_W \sum_i \left(C_vXW_{*i}\right)\cdot\left(C_vZ_{*i}\right) \\
&= \arg\max_W \sum_i \operatorname{cov}\left(XW_{*i}, Z_{*i}\right) \\
&= \arg\min_W \|XW - C_vZ\|_F^2
\end{aligned}$$

Nevertheless, this does not work for the initial optimization objective that maximized the embedding-wise covariance. The reason for this is that, while the column mean centering is preserved after a linear transformation (i.e. $(C_vX)W = C_v(XW)$) and can thus be handled by applying the proposed optimization method over the precomputed matrix C_vX, the row mean centering is not preserved after a linear transformation, since $XC_nW \neq XWC_n$.

7.1.5 Weighted and partial dictionaries

Our discussion so far has assumed that we had a bilingual dictionary of word embeddings (X, Z), but we have not said much about the nature of this dictionary. Even though the conceptually simplest approach would be to limit ourselves to 1:1 correspondences, this would not probably be realistic nor desirable in practice.

Instead of that, we extend our model to incorporate weights for the dictionary entries. This allows to properly model words with different possible translations. For instance, the word “book” is ambiguous in English, as it can be both a noun (bound pages) and a verb (to reserve). Assuming that the distribution is 70-30%, we could therefore have one entry in the dictionary for “book - libro” (bound pages) with a weight of 0.7 and another one for “book - reservar” (to reserve) with a weight of 0.3. In addition to that, weights can also be used for the confidence or frequency of the dictionary entries. For example, if the dictionary is automatically extracted from a word aligned corpus (see Section 2.2), it might be desirable to weight the dictionary entries according to the number of occurrences of the aligned word pairs in the corpus. This would increase the influence of the most frequent equivalences which, intuitively, are more likely to be correct and also more important to get right for most practical applications, as they would also be more likely to occur later on.

Incorporating this weighting schema in the proposed model is straightforward. If we assign a weight of w_i to the ith entry of the dictionary, we simply minimize the weighted sum (or, equivalently, the weighted average) of the squared Euclidean distances according to these weights for the original optimization objective:

$$\arg\min_W \sum_i w_i \|X_{i*}W - Z_{i*}\|^2$$

This is equivalent to multiplying the word embeddings in both languages with the square root of their corresponding weight, so this weighted variant can be solved by simply doing these multiplications in a preprocessing step and then applying the original method over the resulting embeddings:

$$\arg\min_W \sum_i \|\sqrt{w_i}X_{i*}W - \sqrt{w_i}Z_{i*}\|^2$$

It should be noted that this does not only serve for the basic optimization objective in Section 7.1.1, but also for other variants discussed so far. For instance, it can be easily shown that the weighted optimization objective with length normalization and the orthogonality constraint on W maximizes the weighted average cosine similarity between the dictionary entries. For that purpose, it is always necessary to apply the weighting in the last place (after all the normalization steps), since the length normalization would otherwise cancel the effect of the weighting in this case. As an exception, the effect of weighting on dimension-wise mean centering and its relation to average dimension-wise covariance is more subtle. In particular, in the weighted variant mean centering one language is not equivalent to mean centering both depending on when the weighting is applied. We choose to first mean center the embeddings in both languages[4] and then apply the weighting, which

[4]Note that we mean center the monolingual word embeddings and not the columns in the bilingual dictionary matrices, that is, even if an embedding has multiple (or zero) entries in the dictionary, we only count it once

Orthogonality	**Length norm.**	**Mean centering**	**Optimization objective**
			Average squared Euclidean distance
X			Monolingually invariant average squared Euclidean distance
X	X		Monolingually invariant average cosine similarity
X		X	Monolingually invariant average covariance

Table 15: Different optimization objectives within the proposed framework

maximizes the average weighted covariance with unweighted mean centering. Alternatively, it would be possible to first apply the weighting and then the mean centering, which would maximize the average weighted covariance with weighted mean centering. However, we think that our choice is more natural, since the weighting we introduce corresponds to the bilingual correspondence and would not in principle be related to the expected value of the monolingual embeddings in each dimension.

Finally, it is worth mentioning that, just as the proposed model can work with words with several different weighted entries in the dictionary, the learned transformation can also be applied over word embeddings with no entries in the dictionary. In fact, one could reformulate the entire model to eliminate the notion of a strict dictionary and consider all the possible combinations between the embeddings in both languages instead, assigning them a weight analogous to that of the dictionary, which will be 0 for correspondences that are unknown or do not exist in reality. Thanks to this, we are able to naturally incorporate words without any known correspondence into the proposed model.

7.2 Relation to existing bilingual mapping methods

The framework proposed in the previous section starts with a basic optimization objective and introduces several variations that are shown to be equivalent to other optimization goals, which are summarized in Table 15. In this section, we analyze the relationship between our model with all its variations and existing bilingual mapping methods proposed in the literature. In particular, from Section 7.2.1 to 7.2.3 we compare our model to the ones proposed in Mikolov et al. (2013b), Xing et al. (2015) and Faruqui and Dyer (2014), respectively, which were introduced in Section 4.3.1. This serves not only to put our work into context, but also to show the underlying connections between these other methods themselves as part of the general framework we propose. This way, we provide a more global view of bilingual mapping methods that we believe allows for a better understanding of the different models and can motivate new ones in the future.

7.2.1 Mikolov et al. (2013b)

Even though the notation used is slightly different, the optimization objective in Mikolov et al. (2013b) is completely equivalent to the basic one we propose in Section 7.1.1, and they both minimize the average squared Euclidean distance between the dictionary entries. Therefore, our work can be seen as an extension or generalization of it which, based on this basic optimization objective, explores some constraints and normalizations without altering it in any fundamental way, which are shown to be equivalent to other meaningful optimization goals.

The only difference between Mikolov et al. (2013b) and the basic case in our framework is the optimization method used. While we propose an exact method based on the Moore-Penrose pseudoinverse, which can be computed taking the SVD factorization of the source language embedding matrix X, Mikolov et al. (2013b) use stochastic gradient descent instead. We believe that our method has some clear advantages: it is exact (so it is guaranteed to find the optimal solution), fast (it takes less than a second in experimental settings similar to those in Mikolov et al. (2013b)) and can be formulated and reasoned about analytically. As an example of this, it allows to precompute the pseudoinverse X^+, in which case the optimal mapping for any target language embedding matrix Z can be conveniently calculated with a single matrix multiplication $W = X^+Z$. In any case, Mikolov et al. (2013b) do not publish their code nor do they give any detail about its performance, so it is not easy to quantify this in practice. While we would not expect any significant drop in quality or execution time from using stochastic gradient descent (which is in fact likely to be faster for large dictionaries), we think that having an exact method that can be expressed analytically is still desirable.

7.2.2 Xing et al. (2015)

Xing et al. (2015) argue that there are inconsistencies between the optimization objective used to train word embeddings (maximum likelihood based on dot product), the similarity measure typically used for these word embeddings (cosine similarity), and the optimization objective used to learn the linear transformation in Mikolov et al. (2013b) (squared Euclidean distance). Based on this, they propose constraining the word embedding training to make them unit vectors, so that the dot product and cosine similarity become equivalent. In addition to that, they maximize the average dot product (or, equivalently, the cosine similarity) between the dictionary entries instead of minimizing their average squared Euclidean distance to find the best linear transformation between them. So as to guarantee that the word embeddings in the source language will also be unit vectors after applying this mapping, they constrain it to be an orthogonal transformation.

This model is roughly equivalent to our optimization objective with the orthogonality constraint and length normalization presented in Section 7.1.3. However, we next show that our model provides an alternative theoretical justification that we believe is more accurate and conceptually clearer, and also reveals some wrong assumptions in their argumentation. At the same time, we argue that their optimization method is ill-founded and can be quite inefficient, while ours is exact and very fast.

Before getting into that, it should be noted that there is one factor that we leave aside in our discussion, which has to do with the length normalization approach. While Xing et al. (2015) constrain the word embedding training itself to enforce this length normalization, we learn the word embeddings normally and apply the length normalization afterwards[5]. While we do not see a clear reason why their approach should learn better

[5]In principle, this length normalization would only be used to learn the optimal transformation, which

word embeddings as they argue (even if it is true that the word embedding training is based on the dot product and the cosine similarity is typically used instead, it should be noted that this dot product is applied between word vectors and context vectors, and not between the word vectors themselves, and can serve to weight different words depending on their frequency), we consider that this is out of the scope of our work. The method used to normalize the word embeddings is not relevant for our work, as it does not affect the learning of the optimal bilingual mapping in any fundamental way neither from a theoretical nor a practical perspective. This way, the normalization method in Xing et al. (2015) is perfectly compatible with the rest of our model, just as our normalization method is perfectly compatible with the rest of their model, so they can be used and reasoned about interchangeably.

When it comes to the bilingual mapping itself, the first difference between the reasoning of Xing et al. (2015) and ours lies in the motivation for the orthogonality constraint. Xing et al. (2015) argue that "the projected vector Wx_i has to be normalized, which is not guaranteed so far" and propose that "the normalization constraint on word vectors can be satisfied by constraining W as an orthogonal matrix". This way, they incorporate the orthogonality constraint as a way to preserve length normalization. In contrast, our motivation for the same constraint has to do with the wider concept of monolingual invariance: the fact that, after an orthogonal transformation, all the fundamental properties and relations in the original vector space are preserved (not only the length normalization, but also the cosine similarity, the Euclidean distance and the Euclidean norm). This notion of monolingual invariance implies that, if vec("France") is most similar to vec("Spain"), or if vec("France") − vec("Paris") + vec("Germany") is most similar to vec("Berlin") in the original vector space, this should keep being so after the bilingual mapping, and preserving all this fundamental relations, in general, can only be guaranteed by an orthogonal transformation. This way, the orthogonality constraint serves, in the first place, to prevent a degradation in monolingual quality, but also to enforce a relevant property that such transformation should intuitively have, which could be useful to avoid degenerated solutions and learn even better bilingual mappings. All in all, we argue that the rationale for such orthogonality constraint goes far beyond preserving length normalization: while alternatives for preserving length normalization exist (e.g. applying another normalization after the linear transformation as $\frac{Wx_i}{\|Wx_i\|}$), there is a clear motivation for the orthogonality constraint even when no length normalization is performed.

Apart from that, Xing et al. (2015) argue that "the 'closeness' of words in the projection space is measured by the cosine distance, which is fundamentally different from the Euclidean distance in the objective function $\min \sum_i \|Wx_i - z_i\|^2$ and hence causes inconsistence", and they "solve this problem by using the cosine distance in the transform learning". However, we show that this point is completely wrong, since no such inconsistency exists in reality. In Section 7.1.3 we prove that, with the length normalization and orthogonality constraint, minimizing the average squared Euclidean distance is equivalent to maximizing the average cosine similarity. What is more, in Section 7.1.2 we already show

would then be applied to the original embeddings, but it can also be done globally that, even without the length normalization, minimizing the average squared Euclidean distance is equivalent to maximizing the average dot product as long as the transformation is constrained to be orthogonal. Note that the requirement of the orthogonality constraint for such equivalence to hold does not condition the motivation to incorporate such constraint in either work, as preserving the length normalization in their case, or enforcing the monolingual invariance in ours is completely independent from the objective function used.

This way, while Xing et al. (2015) point out that "this inconsistence [between the different objective functions and similarity measures] may lead to suboptimal estimation for both word vectors and the bilingual transform" and consequently "the cosine distance is used [in their work] when we train the orthogonal transform, in order to achieve full consistence", we show that this reasoning is flawed. We prove that the only real difference between their bilingual mapping method and that of Mikolov et al. (2013b) is the length normalization and the orthogonality constraint, and not the objective function used, which is completely equivalent under these conditions. This way, from a theoretical standpoint any gain of Xing et al. (2015) over Mikolov et al. (2013b) can only come from these length normalization and orthogonality constraint they incorporate, and not from solving an inconsistency in the optimization objective for the bilingual mapping, as they argue. Later in Section 7.3, we empirically show that all the improvement actually comes from the orthogonality constraint alone, since the length normalization does not have any clear effect in our experiments. Considering that, as discussed before, the only motivation in Xing et al. (2015) to enforce such orthogonality constraint is precisely preserving the length normalization, we therefore think that our reasoning is not only conceptually clearer, but also more consistent with the observed behavior.

Finally, the optimization method used by Xing et al. (2015) is also different from the one we propose, which we think is better founded, simpler, more efficient and easier to implement. Xing et al. (2015) use a modified version of gradient descent that enforces the orthogonality constraint after each update. More precisely, they compute the gradient as $\nabla W = \sum_i x_i y_i^T$ and update the weights as $W = W + \alpha \nabla W$, where α is the learning rate. After that, they set W to its closest orthogonal matrix:

$$\arg\min_{\bar{W}} \|W - \bar{W}\| \qquad \text{s.t.} \quad \bar{W}^T \bar{W} = I$$

which they solve using the SVD factorization of W. Interestingly, when W is initialized to all-zeros and the gradient is computed over the entire training set, the value that W will take after the first iteration is the exact solution that our method would compute regardless of the learning rate α, since the computations performed in both cases (essentially, taking the SVD factorization of $Z^T X = U \Sigma V^T$ and setting W to VU^T) would be completely equivalent under these conditions. However, this only shows that, under some particular settings, the modified gradient descent finds the optimal solution, but it is not clear what would happen if W were initialized randomly, the gradient were computed over smaller batches, or more iterations were performed. Even if it were proved that the modified gradient descent would still converge to the optimal solution, we think that our method would still have the advantage of being conceptually clearer, the possibility of being expressed analytically, and the fact that it would always be at least as fast as this modified gradient descent, since the simplest possible scenario for it is the one that performs the same computations as our method as discussed previously.

7.2.3 Faruqui and Dyer (2014)

Unlike all the previous methods that attempt to learn the optimal mapping from one language to the other, Faruqui and Dyer (2014) use a well-known statistical technique called canonical correlation analysis (CCA) to project the word embeddings in both languages to a common vector space. More concretely, following our notation CCA finds the vectors a and b so that the Pearson correlation between the projected embeddings $U = Xa$ and $V = Zb$, known as the first pair of canonical variables, is maximal:

$$\arg\max_{a,b} \ \mathrm{corr}(Xa, Zb)$$

After that, CCA finds the second pair of canonical variables, which is the one maximizing the same correlation subject to the constraint that it is uncorrelated with the first pair of canonical variables, and the process is repeated until the desired amount of canonical variables, each of them uncorrelated with the rest, is obtained. This way, if we stack the projection vectors for each language as columns of a wider matrix, we obtain the transformation matrices A and B that project the original word embeddings in their corresponding language to a common vector space. Although by limiting the amount of canonical variables this can also be used to perform dimensionality reduction, as Faruqui and Dyer (2014) themselves explore in their work, we do not consider this factor here to simplify the comparison with our method, and come back to it later in the experiments in Section 7.3.

In order to get better insight into what transformations CCA actually learns, it is convenient to reduce all this to a global optimization objective. The maximization term itself is straightforward to express as the sum of the correlation of all canonical variables $\sum_i \mathrm{corr}\left(XA_{*i}, ZB_{*i}\right)$. As for the constraint that different canonical variables should be uncorrelated among themselves, we take advantage of the fact that the Pearson correlation between two variables is 0 if and only if their covariance is also zero. This way, different canonical variables will only be uncorrelated when their covariance is zero and, therefore, when the covariance matrices of the projected embeddings, which correspond to $\frac{1}{v}(C_vXA)^T(C_vXA) = \frac{1}{v}A^TX^TC_vXA$ and $\frac{1}{v}(C_vZB)^T(C_vZB) = \frac{1}{v}B^TZ^TC_vZB$, are diagonal. Based on this, the global optimization objective of CCA can be expressed as follows:

$$\arg\max_{A,B} \sum_i \mathrm{corr}\left(XA_{*i}, ZB_{*i}\right) \qquad \text{s.t.} \quad \forall i \neq j, \quad (A^TX^TC_vXA)_{ij} = (B^TZ^TC_vZB)_{ij} = 0$$

This expression can be further simplified by taking advantage of the fact that correlation is invariant to the scaling of variables and, therefore, to the scaling of the projection vectors learned by CCA, which correspond to the columns of A and B. This way, it is possible to fix the length of these projection vectors so that the canonical variables have a variance of 1, in which case the Pearson correlation becomes equivalent to the covariance. Furthermore, scaling all these variances by a constant, like the length of the dictionary, would only multiply all covariances by the reciprocal of that constant, and would therefore not affect the global optimization objective. Based on this, the optimization objective of CCA can also be formulated as follows:

$$\arg\max_{A,B} \sum_i \operatorname{cov}\left(XA_{*i}, ZB_{*i}\right) \qquad \text{s.t.} \quad A^TX^TC_vXA = B^TZ^TC_vZB = I$$

This expression clearly resembles the optimization objective of our method with dimension-wise mean centering and the orthogonality constraint discussed in Section 7.1.4:

$$\arg\max_{W} \sum_i \operatorname{cov}\left(XW_{*i}, Z_{*i}\right) \qquad \text{s.t.} \quad W^TW = I$$

As it can be seen, there are only two differences between our optimization objective and that of Faruqui and Dyer (2014): the fact that they learn two linear transformations, one for each language, to project the word embeddings to a common vector space, while we use a single transformation that projects the source language embeddings to the target language, and the constraint imposed on these transformations.

So as to get better insight into these differences, we first analyze what would happen if we introduced an additional orthogonal transformation for the target language embeddings in our optimization objective. For that purpose, it is first convenient to reformulate the original optimization objective as follows:

$$\begin{aligned}\arg\max_{W} \sum_i \operatorname{cov}\left(XW_{*i}, Z_{*i}\right) &= \arg\max_{W} \sum_i \left(C_vXW_{*i}\right) \cdot \left(C_vZ_{*i}\right) \\ &= \arg\max_{W} \sum_i W_{*i}^TX^TC_vZ_{*i} \\ &= \arg\max_{W} \operatorname{Tr}\left(W^TX^TC_vZ\right)\end{aligned}$$

The objective function for the variant with two transformations can be similarly formulated as follows:

$$\begin{aligned}\arg\max_{W} \sum_i \operatorname{cov}\left(XA_{*i}, ZB_{*i}\right) &= \arg\max_{W} \operatorname{Tr}\left(A^TX^TC_vZB\right) \\ &= \arg\max_{W} \operatorname{Tr}\left(BA^TX^TC_vZ\right)\end{aligned}$$

This shows that there is a direct connection between the two variants: when both A and B are constrained to be orthogonal matrices, BA^T will also be an orthogonal matrix, so $BA^T = W^T$. A trivial yet optimal solution will therefore be to take $A = W$ and $B = I$, which clearly shows that transforming both languages instead of a single one does not increase the expressive power of the model under such orthogonality constraint. In fact, it is possible to set B to any arbitrary orthogonal matrix, in which case an optimal solution with $A = WB$ still exists, which is conceptually equivalent to applying our original transformation W to X and then applying such arbitrary (yet orthogonal and, therefore, invariant with respect to all the fundamental properties and relations of the original embeddings as discussed in Section 7.1.2) transformation B to both languages. Moreover, this also applies to the basic unconstrained optimization objective presented in Section 7.1.1 since, when A and B are linear transformations, $W^T = BA^T$ will also be a linear transformation.

This way, the fundamental difference between Faruqui and Dyer (2014) and the proposed model does not lie in the fact that the former transform both languages while we transform a single one, but the constraint imposed on these transformations. In fact, it is precisely because of the chosen constraints that our model cannot benefit from transforming the target language embeddings, whereas CCA does need both transformations to find the optimal solution.

However, a closer look of the constraints shows that they are not that different in reality. In fact, when $X^T C_v X = Z^T C_v Z = I$ the constraints become completely equivalent. This can be extended to any scalar matrix λI for each language independently, since this would only scale the covariance by a constant and would therefore not affect the objective function. Given that $\frac{1}{v} X^T C_v X$ is the covariance matrix of X and $\frac{1}{v} Z^T C_v Z$ is the covariance matrix of Z, what this means is that different dimensions of the word embeddings in each language should have the same variance and be uncorrelated among themselves. Even if the training of word embeddings does not strictly enforce these conditions, they do feel quite intuitive in practice. On the one hand, different dimensions of the word embeddings should intuitively encode different (and not redundant) information, so their correlation should be very low. At the same time, it makes sense that the dispersion in each dimension should be similar, since each dimension should intuitively encode a similar amount of information. Under these assumptions, the conditions in question would hold approximately, so the mapping learned by Faruqui and Dyer (2014) and our method would be similar.

Therefore, both our method and Faruqui and Dyer (2014) coincide on maximizing the average dimension-wise covariance, and the only fundamental difference between them is that, while our model enforces monolingual invariance, Faruqui and Dyer (2014) do change the monolingual embeddings to make different dimensions have the same variance and be uncorrelated among themselves. Even if, as discussed before, these two constraints on the projected embeddings do somehow feel intuitive, we think that mixing both aspects (learning the optimal mapping and enforcing certain properties for the monolingual embeddings) in a single method is not only conceptually confusing, but might also be suboptimal. We consider that, if these properties were actually desirable, the training of the monolingual word embeddings should be adapted to enforce them in the first place[6] and, if this were done, the method of Faruqui and Dyer (2014) and ours would be equivalent for those embeddings. On the contrary, we think that mixing these two aspects could have a negative impact on the learning of the bilingual mapping, and it could also degrade the quality of the monolingual embeddings. Our experiments later in Section 7.3 show empirical evidence supporting this idea.

[6]Alternatively, this might also be done in a postprocessing step

7.3 Experiments on word translation induction

In this section, we experimentally test the proposed framework and all its variants presented in Section 7.1 in comparison with the related methods discussed in Section 7.2. For that purpose, we use the translation induction task introduced by Mikolov et al. (2013b), which learns a bilingual mapping on a small dictionary and measure its accuracy on predicting the translation of new words (see Section 4.4.2). In order to get a more complete picture of the behavior of the different systems, we measure the top 1, top 5 and top 10 accuracies. However, Mikolov et al. (2013b) do not share their dataset and use Google Translate, a commercial online service that is updated continuously, to build their bilingual dictionary, so their experiments cannot be replicated with precision. For that reason, we use the English-Italian dataset on the same task provided by Dinu et al. (2014), which can be freely downloaded from its website[7]. The dataset contains monolingual word embeddings trained with the word2vec toolkit using the CBOW method with negative sampling (see Section 4.1). The context window was set to 5 words, the dimension of the embeddings to 300, the sub-sampling to 1e-05 and the number of negative samples to 10. The English embeddings were trained on a 2.8 billion word corpus (ukWaC + Wikipedia + BNC), while the 1.6 billion word corpus itWaC was used to train the Italian embeddings. In addition to that, the dataset contains a bilingual dictionary learned from Europarl, split into a training set of 5,000 word pairs and a test set of 1,500 word pairs, both of them uniformly distributed in frequency bins of [1-5K], [5K-20K], [20K-50K], [50K-100K] and [100K-200K].

Apart from the performance of the projected embeddings in bilingual terms, we are also interested in the monolingual quality of the source language embeddings after the projection. For that purpose, we use the word analogy task proposed by Mikolov et al. (2013a), which measures the accuracy on answering questions like "what is the word that is similar to *small* in the same sense as *biggest* is similar to *big*?" using simple word vector arithmetic (see Section 4.1). The dataset they use consists of 8,869 semantic and 10,675 syntactic questions of this type, and is publicly available as part of the word2vec toolkit. In order to speed up the experiments, we perform an approximate evaluation by reducing the vocabulary size according to a frequency threshold of 30,000, as suggested by the authors. Since the original embeddings are the same in all the cases and it is only the transformation that is applied to them that changes, this affects all the methods in the exact same way, so the results are perfectly comparable among themselves. With these settings, we obtain a coverage of 64.98%.

We implemented the proposed method in Python using NumPy, an open source package for scientific computing. Following the discussion in Section 7.1, we tested 5 different preprocessing strategies in addition to not performing any preprocessing at all: 1) length normalization (unit), 2) mean centering each target language embedding (trg-word-center), 3) dimension-wise mean centering (dim-center), 4) length normalization followed by mean centering each target language embedding (unit + trg-word-center) and 5) length normalization followed by dimension-wise mean centering (unit + dim-center). In all the cases,

[7]http://clic.cimec.unitn.it/%7Egeorgiana.dinu/down/

we tested the unconstrained variant and the variant with the orthogonality constraint and, when relevant, we also tested applying the preprocessing globally (i.e. not only learning the bilingual mapping over the normalized embeddings but also applying it to these normalized embeddings instead of the original ones). As for the method by Faruqui and Dyer (2014), we used their original implementation in Python and MATLAB, which is publicly available at GitHub[8]. However, this code does not account for cases where the dictionary contains more than one entry for the same word, as the dataset we use does, so we modified it accordingly to fix this issue. As mentioned in Section 7.2.3, Faruqui and Dyer (2014) also perform dimensionality reduction when applying CCA, which we also tested in steps of 10%. In addition to that, even if it is not described in the original paper, their implementation allows to learn a single transformation to map the embeddings in one language to the other instead of learning two transformations to project them to a shared vector space, and we also tested this variant. Following the notation in Section 7.2.3, they simply set $W = AB^{-1}$ for that so, in this case, no dimensionality reduction can be performed. Finally, the code for Mikolov et al. (2013b) and Xing et al. (2015) is not publicly available, so we implemented and tested them as part of the proposed framework, which only differs from the original systems in the optimization method (exact solution instead of gradient descent) and the length normalization approach in the case of Xing et al. (2015) (postprocessing instead of constrained training).

The obtained results are shown in Table 16. As it can be seen, the method by Xing et al. (2015) performs consistently better than that of Mikolov et al. (2013b) in the translation induction task, which is in line with what they report in their paper. Moreover, thanks to the orthogonality constraint their monolingual performance in the word analogy task does not degrade, whereas the accuracy of Mikolov et al. (2013b) drops by 2.86% in absolute terms with respect to the original embeddings. However, it is remarkable that several configurations of Faruqui and Dyer (2014) beat the method by Xing et al. (2015) in the bilingual task, although this comes at the price of a considerable degradation in monolingual quality. More concretely, when the embeddings in both languages are projected to a shared space, the bilingual performance improves as the dimensionality is reduced up to around 50%, where it starts to degrade. The monolingual performance is harmed even when no dimensionality reduction is applied, in particular for the syntactic analogies (an absolute drop of 6.95% with respect to the original embeddings in contrast with 1.67% for the semantic analogies), and is further accentuated as the dimension of the embeddings is reduced, in particular for the semantic analogies (an absolute drop of 9.26% when going from 100% to 50% in contrast with only 1.35% for the syntactic analogies). In comparison, the direct mapping variant does not affect monolingual quality as much (an absolute drop of 2.39% with respect to the original embeddings, less than the 2.86% of Mikolov et al. (2013b)), and it also achieves a better top 1 accuracy with 38.7%, an improvement of 1.8% with respect to Xing et al. (2015) and 3.8% with respect to Mikolov et al. (2013b). However, the top 5 and top 10 accuracies are slightly worse than when applying a dimensionality reduction of 50% with the shared space variant, but still better than those of Xing

[8]https://github.com/mfaruqui/crosslingual-cca

et al. (2015) and Mikolov et al. (2013b).

In any case, it is the proposed method with the orthogonality constraint and a global preprocessing with length normalization followed by dimension-wise mean centering that achieves the best top 1, top 5 and top 10 accuracies in the word translation induction task. Moreover, it does not suffer from any considerable degradation in monolingual quality, with an anecdotal drop of only 0.07% with respect to the original embeddings, in contrast with more than 2% for Mikolov et al. (2013b) and Faruqui and Dyer (2014). Therefore, it can be said that it is this variant of the proposed method that gives the best overall results, as it clearly offers the best bilingual performance along with a monolingual performance at par with Xing et al. (2015) and considerably superior to Mikolov et al. (2013b) and Faruqui and Dyer (2014).

A closer look at the different variants of the proposed model reveals that all the configurations with the orthogonality constraint get better results than their unconstrained counterparts for all the metrics considered, without any single exception. This is a very strong evidence in favor of our hypothesis that the orthogonality constraint does not only prevent a degradation in monolingual performance, but it also improves bilingual performance by enforcing a relevant property (monolingual invariance) that the transformation to learn should intuitively have.

As for the different preprocessing configurations tested, we first observe that length normalization alone does not prove to be beneficial. When applied together with the orthogonality constraint, it improves the top 1 accuracy in the translation induction task by only 0.2%, while degrading the top 5 and top 10 accuracies by 0.1% and 0.7%, respectively. Together with the previous point, this reinforces our theoretical interpretation in Section 7.2.2 for the method by Xing et al. (2015), as it empirically shows that its improvement with respect to Mikolov et al. (2013b) comes solely from the orthogonality constraint, and not from solving any kind of inconsistency.

Apart from that, we observe that mean centering each target language embedding for embedding-wise covariance maximization does not bring any improvement either. However, dimension-wise mean centering for dimension-wise covariance maximization does prove to be beneficial. Moreover, when this preprocessing is applied globally, preceding this dimension-wise mean centering by length normalization further improves bilingual performance, achieving the best overall results as discussed above. Interestingly, this is the only instance where either applying the normalization globally or preceding the mean centering by length normalization yields to a clear improvement in the word translation induction task.

It should be noted that, even if it is not mentioned in the paper, the implementation by Faruqui and Dyer (2014) also length normalizes the word embeddings in a preprocessing step. Following the discussion in Section 7.2.3, this means that our best performing configuration is conceptually very close to the method by Faruqui and Dyer (2014), as they both coincide on maximizing the average dimension-wise covariance and length normalize the embeddings in both languages first, the only difference being that our model enforces monolingual invariance after the normalization while theirs does change the monolingual embeddings to make different dimensions have the same variance and be uncorrelated

		EN-IT TRANSLATION (bilingual performance)			EN ANALOGIES (monolingual performance)		
		P@1	**P@5**	**P@10**	**Sem.**	**Synt.**	**Total**
Original embeddings		-	-	-	**79.66%**	**75.36%**	**76.66%**
Mikolov et al. (2013b)*		**34.9%**	**49.1%**	**54.5%**	**73.07%**	**74.12%**	**73.80%**
Xing et al. (2015)*		**36.9%**	**52.7%**	**57.9%**	**79.66%**	**75.36%**	**76.66%**
Faruqui and Dyer (2014) (shared space)	n=30 (10%)	24.7%	39.7%	44.3%	22.24%	39.98%	34.61%
	n=60 (20%)	34.3%	49.7%	55.2%	43.42%	57.58%	53.30%
	n=90 (30%)	36.6%	53.6%	58.7%	58.43%	64.66%	62.78%
	n=120 (40%)	37.8%	55.0%	60.9%	66.13%	66.15%	66.15%
	n=150 (50%)	37.4%	**55.5%**	**61.4%**	68.73%	67.06%	67.56%
	n=180 (60%)	37.8%	54.4%	59.3%	72.06%	68.58%	69.64%
	n=210 (70%)	37.4%	54.1%	59.0%	74.53%	68.74%	70.49%
	n=240 (80%)	35.8%	53.5%	58.5%	75.83%	68.91%	71.01%
	n=270 (90%)	34.5%	50.7%	56.9%	75.70%	68.80%	70.89%
	n=300 (100%)	32.4%	49.1%	54.1%	**77.99%**	68.41%	71.31%
Faruqui and Dyer (2014) (direct mapping)		**38.7%**	55.1%	60.5%	77.86%	**72.72%**	**74.27%**
Proposed method (unconstrained)	-	34.9%	49.1%	54.5%	73.07%	74.12%	73.80%
	unit	33.8%	48.3%	53.9%	72.92%	73.91%	73.61%
	trg-word-center	34.8%	49.1%	54.1%	72.87%	74.15%	73.76%
	dim-center	37.3%	51.9%	56.3%	73.13%	74.33%	73.97%
	unit + trg-word-center	34.1%	48.1%	53.7%	72.97%	73.98%	73.68%
	unit + dim-center	30.4%	45.3%	49.4%	72.94%	73.70%	73.47%
Proposed method (orthogonal transform)	-	36.7%	52.8%	58.6%	**79.66%**	75.36%	76.66%
	unit	36.9%	52.7%	57.9%	**79.66%**	75.36%	76.66%
	trg-word-center	36.8%	53.0%	58.1%	**79.66%**	75.36%	76.66%
	dim-center	38.0%	54.8%	59.8%	**79.66%**	75.36%	76.66%
	unit + trg-word-center	36.8%	52.7%	58.2%	**79.66%**	75.36%	76.66%
	unit + dim-center	37.1%	53.5%	58.4%	**79.66%**	75.36%	76.66%
Proposed method (unconstrained + global normalization)	trg-word-center	34.7%	49.1%	54.2%	72.87%	74.15%	73.76%
	dim-center	37.8%	53.4%	59.1%	73.49%	74.42%	74.14%
	unit + trg-word-center	33.8%	48.3%	54.0%	72.97%	73.98%	73.68%
	unit + dim-center	38.5%	55.9%	60.5%	73.07%	73.99%	73.71%
Proposed method (orthogonal transform + global normalization)	trg-word-center	36.7%	52.8%	58.4%	**79.66%**	75.36%	76.66%
	dim-center	37.9%	54.9%	60.5%	79.47%	**75.49%**	**76.70%**
	unit + trg-word-center	37.1%	52.6%	58.1%	**79.66%**	75.36%	76.66%
	unit + dim-center	**39.3%**	**56.3%**	**61.7%**	79.63%	75.27%	76.59%

Table 16: Results on English-Italian word translation induction. The best results for each method (denoted by blocks separated by double lines) are given in bold, and the best overall results are underlined. Mikolov et al. (2013b) and Xing et al. (2015) are implemented as part of the proposed framework, using our length normalization and exact optimization methods.

among themselves. However, our model performs considerably better than any configuration from Faruqui and Dyer (2014) in both the monolingual and the bilingual task, supporting our hypothesis in Section 7.2.3 that these two constraints that are implicit in their method are not only conceptually confusing, but also have a negative impact.

To sum up, our experiments show the effectiveness of the proposed model and give strong empirical evidence in favor of our theoretical reasoning in Section 7.2. It is the proposed method with the orthogonality constraint and a global preprocessing with length normalization and dimension-wise mean centering that achieves the best overall results both in monolingual and bilingual terms, clearly surpassing the previous methods by Mikolov et al. (2013b), Xing et al. (2015) and Faruqui and Dyer (2014). Moreover, the obtained results support our alternative interpretation for Xing et al. (2015) and Faruqui and Dyer (2014) in relation to the proposed framework.

7.4 Conclusions

In this chapter, we have proposed a general framework to learn bilingual word embedding mappings. It starts with a basic optimization objective that is equivalent to the previous model by Mikolov et al. (2013b) and allows for several variants that we prove to be equivalent to other meaningful and relevant optimization objectives in close relation to the previous methods by Xing et al. (2015) and Faruqui and Dyer (2014). This way, our framework provides a more global view of bilingual word embedding mappings, showing the underlying connection between the existing methods, revealing some flaws in their theoretical justification and providing an alternative theoretical interpretation for them. Our experiments on an existing English-Italian word translation induction and an English word analogy task give strong empirical evidence in favor of our theoretical reasoning, while showing that a variant of the proposed model clearly outperforms all the previous methods tested. More concretely, our model achieves the best bilingual performance in the word induction task, while preventing a degradation in monolingual quality and consequently performing at par with Xing et al. (2015) and considerably better than the rest of the methods in the monolingual word analogy task.

8 Bilingual word embeddings for phrase translation similarity scoring

In this chapter, we explore the use of bilingual word embeddings for phrase translation similarity scoring. Unlike many of the previous approaches discussed in Section 4.4.1, which train bilingual phrase embeddings from scratch using complex architectures, we use different metrics to directly compute a phrase similarity score over bilingual word embeddings. A simple yet widely used measure in this regard is the centroid cosine similarity, which we analyze from a theoretical perspective in Section 8.1. Section 8.2 then proposes three alternative measures that address some of the issues of centroid cosine similarity as pointed by our analysis. In Section 8.3, we experimentally test the proposed measures with different bilingual word embedding methods, including our bilingual mapping model in Chapter 7, on English-Spanish phrase translation selection. Finally, Section 8.4 applies them to end-to-end English-Spanish machine translation, and Section 8.5 concludes the chapter.

8.1 Analysis of centroid cosine similarity

As discussed in Section 4.1, the cosine similarity is typically used to measure the semantic similarity of two words from their embeddings. Recall that, given two vectors x and y, their cosine similarity $\cos(x, y)$ is given by their normalized dot product:

$$\cos(x, y) = \frac{x \cdot y}{\|x\|\|y\|}$$

However, it is not trivial to generalize this to measure the semantic similarity of larger phrases. As discussed in Section 4.4.1 for the specific field of MT, there is a considerable research effort on learning embedding representations for the phrases themselves, which mostly rely on recurrent or recursive neural networks that compose individual word vectors iteratively. Nevertheless, these techniques are often task specific, tend to use complex architectures that are expensive to train, and typically learn not only the phrase embeddings themselves, but also the underlying word embeddings from scratch.

For that reason, a simple yet widely used approach is to take the centroid of the word embeddings in each phrase and use the cosine similarity over these centroids as a measure of their semantic similarity. These centroids can be seen as continuous bag-of-words representations of phrases, which in fact gives its name to the CBOW architecture to learn word embeddings as seen in Section 4.1. In spite of the popularity of this technique, the relation between the centroid cosine similarity and the underlying word embeddings is not obvious, and works using it in the literature do not generally give any insight in this regard. This way, we next analyze what the centroid cosine similarity actually computes.

Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ be the embeddings of the words in two phrases. Their centroid or arithmetic mean is given by $\frac{1}{n}\sum_{i=1}^{n} x_i$ and $\frac{1}{m}\sum_{i=1}^{m} y_i$, respectively, so the cosine similarity between them corresponds to the following:

$$\begin{aligned}\cos\left(\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{m}\sum_{i=1}^{m} y_i\right) &= \frac{\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right)\cdot\left(\frac{1}{m}\sum_{i=1}^{m} y_i\right)}{\|\frac{1}{n}\sum_{i=1}^{n} x_i\|\|\frac{1}{m}\sum_{i=1}^{m} y_i\|}\\ &= \frac{\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} x_i\cdot y_j}{\sqrt{\frac{1}{n^2}\left(\sum_{i=1}^{n} x_i\right)\cdot\left(\sum_{i=1}^{n} x_i\right)}\sqrt{\frac{1}{m^2}\left(\sum_{i=1}^{m} y_i\right)\cdot\left(\sum_{i=1}^{m} y_i\right)}}\\ &= \frac{\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} x_i\cdot y_j}{\sqrt{\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i\cdot x_j}\sqrt{\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i\cdot y_j}}\end{aligned}$$

As it can be seen, the numerator in the last expression, which also corresponds to the dot product of the centroids, is the average dot product of all the word combinations between both phrases. The denominator is the square root of the average dot product of all the word combinations within each phrase, and serves to normalize it so that it takes a value between -1 and 1. Since, from the definition of the cosine similarity, $x\cdot y = \|x\|\|y\|\cos(x,y)$, this can be rewritten as follows:

$$\frac{\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\|x_i\|\|y_j\|\cos(x_i,y_j)}{\sqrt{\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|x_i\|\|x_j\|\cos(x_i,x_j)}\sqrt{\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\|y_i\|\|y_j\|\cos(y_i,y_j)}}$$

Therefore, the centroid dot product can also be seen as the average cosine similarity of all the word combinations between both phrases multiplied by their length, so that longer vectors have more influence than shorter ones. Similarly, the normalization term in the denominator corresponds to the square root of the average cosine similarity of all the word combinations within each phrase multiplied by their length.

It is particularly interesting to see what happens when the word embeddings are normalized, that is, $\forall i, j, \|x_i\| = \|y_j\| = 1$, which can be done by dividing all the vectors by their actual norm. This normalization is quite common, as the length of word embeddings does not generally carry any clear meaning (in fact, the cosine similarity itself is independent from the length of the vectors and can be seen as a normalized dot product). In this case, the cosine similarity of two phrase centroids corresponds to the following:

$$\cos\left(\frac{1}{n}\sum_{i=1}^{n}\frac{x_i}{\|x_i\|}, \frac{1}{m}\sum_{i=1}^{m}\frac{y_i}{\|y_i\|}\right) = \frac{\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\cos(x_i,y_j)}{\sqrt{\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\cos(x_i,x_j)}\sqrt{\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\cos(y_i,y_j)}}$$

Consequently, the dot product of the centroids of two phrases with normalized word embeddings, which corresponds to the numerator in the above expression, is the average cosine similarity of all the word combinations between both phrases. At the same time, the Euclidean norm of the centroid of a phrase with normalized word embeddings, which corresponds to the two factors in the denominator above, is the square root of the average cosine similarity of all the word combinations within that phrase. Intuitively, the centroid cosine similarity thus measures the similarity between two phrases according to the average similarity of all the word combinations between them, and then normalizes this value by compensating phrases with very different words within them.

Even though this clearly explains why and how the centroid cosine similarity works and its relation with the underlying word embeddings, it also raises a question of how appropriate it is. Intuitively, we would asses the similarity between “a famous artist” and “two prestigious painters”, for example, according to the similarity between “a” and “two”, “famous” and “prestigious”, and “artist” and “painters”. It does not feel intuitive to also consider, as the centroid cosine similarity would, the similarity between all the other word combinations between and within the phrases such as “famous” and “painters”, “a” and “artist”, or “two” and “famous”, and even less to give the same importance to all of them or to weight them according to the length of their word embeddings, which does not carry any clear meaning. For that reason, in the next subsection alternative word embedding based phrase similarity measures that are closer to the human intuition are proposed.

8.2 Proposed phrase similarity measures

As seen in the previous section, in spite of the popularity of the centroid cosine similarity, a closer analysis of it reveals that what it actually computes has little to do with how humans would measure phrase similarity based on word similarity. For that reason, we propose the following phrase similarity measures that we believe better capture the human intuition:

- **Arithmetic mean of closest word cosine similarity**. As seen before, the centroid cosine similarity of two phrases is essentially a normalized average of the cosine similarity of all the word combinations between both phrases. However, a human would intuitively consider each word in relation with its semantic counterpart in the other phrase, if any. Even though we do not in principle have any prior alignment, it feels intuitive that each word would most likely be related to the one that is most similar to it in the other phrase. Based on this, we can estimate phrase similarity by taking the average cosine similarity between each word and its closest counterpart in the other phrase as measured by cosine similarity itself:

$$\frac{\sum_{i=1}^{n} \max_j \cos(x_i, y_j) + \sum_{i=1}^{m} \max_j \cos(y_i, x_j)}{n+m}$$

  This is in line with the measure proposed by Mihalcea et al. (2006) to combine word-to-word similarity scores for text semantic similarity scoring, with the difference that we take an unweighted average and specifically adapt the measure for cosine similarity. It should be noted that, even though it might not seem obvious at first, it is actually necessary to take the closest word in both directions as in the above formula to account for cases where there is missing or additional information in one of the phrases. More concretely, if only one direction were considered, the target phrase having extra words with no counterpart in the source phrase would not have any negative effect, whereas the target phrase missing information would be heavily penalized. By considering both directions, we overcome this issue and preserve the symmetry of the similarity measure.

- **Product of closest word normalized cosine similarity**. Levy and Goldberg (2014a) point out that, when taking the sum of different cosine similarities, one sufficiently large term tends to dominate the result, which they consider problematic in their setting where each word embedding represents a different aspect of similarity in its own scale. For instance, they argue that "king" is more royal than masculine, which might overshadow the gender aspect in their similarity measure. Based on this, they propose taking the product of the different cosine similarities instead of their sum, obtaining a considerable improvement in an analogy task.

  Even if we are dealing with a quite different task, a similar issue might also arise when taking the arithmetic mean of different cosine similarities as the previous measure or the centroid cosine similarity itself do. For example, "the king said something" might be considered more similar to "the parrot said something" than "the man mentioned something" since, assuming that the royalty aspect is indeed central to "king", both "man" and "parrot" could be very far from it, possibly making the absolute difference between their similarities insufficient to compensate the nuance between "said" and "mentioned" (e.g. if $\cos(king, man) = 0.2$, $\cos(king, parrot) = 0.1$ and $\cos(said, mentioned) = 0.8$).

  Based on this, we can take the product of the cosine similarity between each word and its closest counterpart in the other phrase instead of their arithmetic mean. However, this would be problematic if negative cosine similarities were involved. For that reason, instead of directly taking the cosine similarity we normalize its value to $[0, 1]$:

$$\prod_{i=1}^{n} \max_j \frac{1 + \cos(x_i, y_j)}{2} \prod_{i=1}^{m} \max_j \frac{1 + \cos(y_i, x_j)}{2}$$

- **Product of closest word thresholded cosine similarity**. The previous measure normalized cosine similarity to $[0, 1]$ to avoid negative values in the multiplication. However, it should be noted that the expected cosine similarity between two random vectors is 0, so one can expect that this will rarely occur in practice: even if a given word has no direct counterpart in the other phrase, odds are that the cosine similarity with some word on it is still greater than 0. In addition to that, a very small closest word similarity is likely to indicate that the information in question is not present in the other phrase, so the exact value it takes is arguably irrelevant. For that reason, an alternative to normalizing cosine similarities is to take a small positive threshold ϵ so that all cosine similarities below it are replaced with it:

$$\prod_{i=1}^{n} \max(\epsilon, \max_j \cos(x_i, y_j)) \prod_{i=1}^{m} \max(\epsilon, \max_j \cos(y_i, x_j))$$

It should be noted that these similarity measures behave differently when it comes to the length of the phrases. For the first measure, each term can have a negative or positive impact in the final score with respect to the others, so different length phrase pairs are scored in the same scale. As a consequence, “good car” and “excellent book” would get a higher similarity score than “car” and “book” alone, since the semantic closeness between “good” and “excellent” would make up for the big distance between “car” and “book”. In contrast, for the other two measures each term can only have a negative impact in the final score[9], so longer phrase pairs will tend to have lower similarities. As a consequence, “car” and “book” would get a higher similarity score than “good car” and “excellent book”, since the difference between “good” and “excellent”, although minor, would further increase the distance between both phrases.

Needless to say, the appropriateness of one behavior or the other will depend on the specific application, and it would mostly depend on the interaction between different length phrase pairs, if any. In the case of SMT, this would mostly affect segmentation, and not that much the ranking of the different translation candidates for a given phrase. In this regard, the first approach is in line with what translation probabilities do, as each phrase contributes equally to the total similarity score of a translation candidate regardless of its length, presumably favoring long phrase segmentations, whereas the second approach is in line with what lexical weightings do, as each word contributes equally to the total similarity score regardless of the segmentation.

Finally, it should be noted that the different similarity measures proposed can be easily adapted to invert their behavior with respect to the length of the phrases by taking their $(n + m)$th power or root, depending on the case. For the last two measures, this would be equivalent to taking the geometric mean (as opposed to the product) of the closest word cosine similarities. However, we did not observe any clear improvement from doing that in our preliminary experiments.

8.3 Experiments on phrase translation selection

Even if the ultimate goal of our work is to improve machine translation, current SMT systems are made of complex components interacting in subtle ways (see Section 2.3), so it might be hard to extract clear conclusions on the behavior of different bilingual word embeddings and phrase similarity measures from an extrinsic evaluation there (e.g. due to the stochastic tunning process or the incorporation of possibly redundant information). For that reason, we leave the evaluation on end-to-end machine translation for the next subsection and focus on the intrinsic evaluation of the different bilingual word embeddings and phrase similarity measures in this one.

There are several experimental frameworks that could be used for that purpose, as it is the case of the new crosslingual semantic textual similarity task introduced at the International Workshop on Semantic Evaluation 2016 (SemEval-2016)[10]. However, we instead

[9]This only applies to information present in both languages since, as discussed before, all the proposed similarity measures do penalize missing information in either phrase

[10]http://alt.qcri.org/semeval2016/task1/

propose a new task on phrase translation selection, which has the advantage of relying solely on unsupervised data, allowing much larger test sets for more robust evaluation at practically no cost and, in our case, being tightly bound to machine translation, while isolated from its different components.

Our task needs nothing but a phrase table from a standard phrase-based SMT system and a word aligned parallel test set. In our case, we use the alignment model learned from the parallel corpora used to train the SMT system to word align the test set. Having done that, we extract all the aligned phrase pairs in the test set that are found in the phrase table, in a similar way to how we create examples to train our logistic regression model in Section 5.1. Finally, we use the similarity measure and bilingual word embeddings under test to score the different translation candidates in the phrase table, and measure the top 1, top 5 and top 10 accuracies on predicting the translations for the extracted phrase pairs.

For our experiments, we reused the phrase table from the English-Spanish baseline system in Section 5.4 and take the WMT12 corpus presented there as our test set. Note that, even if this was our development set there, it was exclusively used for tuning, which does not affect the creation of the phrase table, so there is no problem on using it as our test set here. This way, we keep the WMT13 corpus for the test set in the actual end-to-end machine translation experiments later in Section 8.4, allowing a fair comparison with respect to the baseline system.

In addition to randomly selecting the target language phrase from all the candidates and using each of the weights in the phrase table (inverse and forward translation probabilities and lexical weightings), we test several different bilingual word embedding methods presented in Section 4.3 under these conditions. We first implemented the simple Align-Init method proposed by Zou et al. (2013), learning monolingual phrase embeddings for the source language with word2vec and initializing the target language ones according to the forward lexical weights given by word alignment (see Section 2.2). Even if their full model runs a bilingual training procedure over the embeddings initialized this way, we skip this part and directly use the alignment initialized target language embeddings, which the authors report to give slightly worse yet competitive results in machine translation (an improvement of 0.30 BLEU points compared to the 0.49 BLEU points of the full model). In addition to that, we also tested the BilBOWA model by Gouws et al. (2014) and the BiSkip model by Luong et al. (2015), using the original open source implementations by the authors at GitHub[11]. In the case of BiSkip, the authors report that they take the sum of the input and output (i.e. word and context) vectors instead of the input ones alone as it is commonly done, so we tested both variants. Finally, we also test the best performing bilingual mapping model in our translation induction experiments in Section 7.3, which happens to be the model we propose in Section 7.1 with the orthogonality constraint and a global preprocessing with length normalization and dimension-wise mean centering.

In all the cases, we trained the word embeddings using the skip-gram model, and we set the context window to 5, the dimension of the embeddings to 300, the subsampling to 1e-4, the number of negative samples to 30 and the number of iterations to 10 (see Section

[11]https://github.com/gouwsmeister/bilbowa and https://github.com/lmthang/bivec

	Domain	Sentences	English tokens	Spanish tokens
Europarl	parliamentary proceedings	1,929,366	51,593,389	54,021,555
News 2007-2012 (es)	news	13,384,314	-	386,015,256
News 2007-2012 (en, balanced)	news	13,384,314	388,443,422	-
News 2007-2012 (en, full)	news	68,521,621	1,612,954,315	-

Table 17: Corpora for training bilingual word embeddings

4.1). Default values were used for all the other parameters.

So as to allow a fair comparison with respect to the baseline SMT system, we used the same parallel corpora (Europarl) to train the word embeddings. In addition to that, we also tested incorporating the News 2007-2012 monolingual corpora. Since the English portion of it is considerably larger than the Spanish one, we took a balanced subset of it at random when relevant. Table 17 summarizes their details. Our vocabulary consisted of all the words in the bilingual corpus plus, in case any monolingual corpora was used, all the words with at least 5 occurrences in total. In the case of BilBOWA, we were unable to perform any experiment over the monolingual corpora because of the slow training process. For Align-Init, other than to train the monolingual source language embeddings the bilingual corpus was also used to extract the forward lexical weights to initialize the target language embeddings. Similarly, in the case of our proposed method we used the symmetrized word alignments from the bilingual corpus to build our dictionary, weighting the entries according to the number of times each word pair was aligned (see Section 7.1.5).

For each bilingual word embedding method and training corpus combination, we tested the different phrase similarity measures discussed throughout the chapter: the centroid cosine similarity, the length normalized centroid cosine similarity, the arithmetic mean of closest word cosine similarity, the product of closest word normalized cosine similarity and the product of closest word thresholded cosine similarity. In the last case, after a preliminary exploration we set the threshold ϵ to 0.1.

The obtained results are given in Table 18. It is first remarkable that, among all the similarity measures tested, it is the closest word cosine product we propose which, in its two variants (normalized and thresholded), achieves the best results by a large margin, beating the other metrics for all the corpora and bilingual word embedding methods tested without any single exception. The improvement with respect to the standard centroid cosine similarity is particularly remarkable in some cases. For instance, in the case of the BiSkip word vectors trained on Europarl, the closest word thresholded cosine product achieves a top 1 accuracy of 56.96% in contrast with the 14.83% of the centroid cosine similarity, an absolute improvement of 42,13%. The improvement is logically more modest in the general case, but still very significant, in particular considering that the bilingual word embeddings are the same in all the cases and it is only the phrase similarity measure computed over them that changes. For example, the best top 1 accuracy for the centroid cosine similarity across all the configurations tested, obtained by its normalized variant, is 55.38%, whereas the thresholded version of the closest word cosine product is able to achieve a top 1 accuracy of 60.41%, an absolute improvement of 5.03%. As for the two variants of the said metric, we observe that the thresholded version tends to give better results for the top 1 accuracy while the normalized version tends to perform better in the top 5 and top 10 accuracies, but there are exceptions to this trend and the results are not conclusive in this regard. In any case, we consider that the experiments clearly show the important contribution of the closest word cosine similarity measure we propose in either of its variants with respect to the commonly used centroid cosine similarity. The arithmetic mean of the closest word cosine similarity is somewhat in between: while it performs considerably better than the centroid cosine, with the only exception of Align-Init, its results are clearly below those of the closest word cosine product.

As far as the different word embedding methods are concerned, we first observe that BilBOWA gives the worst results by a large margin. Considering that its training times are also the highest by far, we conclude that this is undoubtedly the worst performing method across the ones we test, at least for this task. Apart from that, we observe that both variants of BiSkip and the method we propose give better results than Align-Init when it comes to the best performing measure and training corpus in each case. It is remarkable that the centroid cosine similarity gives comparatively better results in the case of Align-Init, but still not as good as those of the closest word cosine product. As for the two variants of the BiSkip model, we observe that using the sum of the word and context vectors, as the authors reported, gives better results in general. We think that this is an interesting finding, since this simple technique could in principle be applied to any word embedding method. In any case, it is the proposed method discussed in Section 7 that achieves the best top 1 accuracy among all the bilingual embedding models tested, with an absolute gain of 2.31% with respect to BiSkip, the second best performing method, for their best configuration. However, we observe that BiSkip gets better top 5 and top 10 accuracies. This suggests that, while our method is clearly the best at getting its first choice right, the relative order for the rest of the candidates given by BiSkip might be better.

Another interesting variable to analyze is the effect of the training corpus. Contrary to our expectations, we observe that using an additional monolingual corpus to train the bilingual embeddings has a negative impact in many cases. This is clearly the case of both variants of BiSkip, although it should be noted that the method was not originally conceived to work with monolingual corpora, which might explain the bad results. In the case of Align-Init, using a monolingual corpus also has a negative effect in the top 1 accuracy, although it obtains a similar top 5 accuracy and a slightly better top 10 accuracy for the closest word cosine product. In contrast, the proposed bilingual mapping method clearly benefits from using an additional monolingual corpus. For instance, in the case of the closest word thresholded cosine product it gets an absolute improvement of 0.91%, 1.99% and 0.43% in the top 1, top 5 and top 10 accuracies when using the full monolingual corpus. We think that this is a very interesting property that the proposed method has, as it allows to train better bilingual word embeddings by incorporating distributional information from monolingual corpora. This is particularly interesting for machine translation since, even if target language monolingual corpora is commonly used for language modeling, current SMT models cannot benefit from source language monolingual corpora.

			P@1	P@5	P@10
Random selection	-	-	**10.05%**	**34.36%**	**58.30%**
		Inverse translation probability	43.27%	78.45%	90.14%
Phrase-based SMT	*Europarl*	Inverse lexical weighting	14.52%	43.65%	67.22%
		Forward translation probability	**64.47%**	**94.71%**	**98.58%**
		Forward lexical weighting	57.45%	91.99%	98.08%
		Centroid cosine	53.86%	81.98%	90.10%
		Length normalized centroid cosine	54.59%	83.84%	91.85%
	Europarl	Closest cosine arithmetic mean	53.01%	83.35%	92.01%
Align-Init		Closest cosine product (normalized)	**55.77%**	88.43%	96.07%
forward lexical weight init		Closest cosine product ($\epsilon = 0.1$)	55.42%	88.39%	96.07%
		Centroid cosine	50.05%	81.37%	90.33%
(Zou et al., 2013)	*Europarl*	Length normalized centroid cosine	50.02%	82.14%	88.89%
	+	Closest cosine arithmetic mean	48.87%	81.29%	88.77%
	Full news	Closest cosine product (normalized)	53.13%	**88.49%**	**96.60%**
		Closest cosine product ($\epsilon = 0.1$)	51.59%	88.12%	**96.60%**
		Centroid cosine	4.87%	17.51%	26.25%
BilBOWA		Length normalized centroid cosine	4.89%	17.47%	26.13%
	Europarl	Closest cosine arithmetic mean	13.65%	30.64%	49.84%
(Gouws et al., 2014)		Closest cosine product (normalized)	**30.24%**	**60.32%**	**77.75%**
		Closest cosine product ($\epsilon = 0.1$)	30.21%	60.22%	77.57%
		Centroid cosine	14.83%	37.96%	59.34%
		Length normalized centroid cosine	21.50%	42.39%	60.61%
	Europarl	Closest cosine arithmetic mean	49.63%	75.87%	86.75%
BiSkip		Closest cosine product (normalized)	55.69%	**91.38%**	**98.03%**
word vectors		Closest cosine product ($\epsilon = 0.1$)	**56.96%**	91.06%	97.69%
		Centroid cosine	4.81%	16.51%	31.62%
(Luong et al., 2015)	*Europarl*	Length normalized centroid cosine	5.06%	15.53%	29.17%
	+	Closest cosine arithmetic mean	11.09%	29.99%	57.95%
	Balanced news	Closest cosine product (normalized)	40.65%	79.27%	94.86%
		Closest cosine product ($\epsilon = 0.1$)	40.53%	78.51%	94.03%
		Centroid cosine	31.44%	62.31%	74.50%
		Length normalized centroid cosine	34.99%	63.75%	74.64%
	Europarl	Closest cosine arithmetic mean	53.78%	77.89%	89.08%
BiSkip		Closest cosine product (normalized)	57.30%	**91.38%**	**97.98%**
word + context vectors		Closest cosine product ($\epsilon = 0.1$)	**58.10%**	90.26%	96.70%
		Centroid cosine	4.86%	20.77%	33.34%
(Luong et al., 2015)	*Europarl*	Length normalized centroid cosine	5.06%	19.53%	34.94%
	+	Closest cosine arithmetic mean	10.11%	28.69%	50.18%
	Balanced news	Closest cosine product (normalized)	39.92%	77.83%	94.49%
		Closest cosine product ($\epsilon = 0.1$)	39.52%	76.95%	92.88%
		Centroid cosine	50.32%	71.50%	82.09%
		Length normalized centroid cosine	51.01%	72.33%	80.84%
	Europarl	Closest cosine arithmetic mean	56.16%	80.23%	87.93%
		Closest cosine product (normalized)	59.37%	86.96%	95.89%
		Closest cosine product ($\epsilon = 0.1$)	59.50%	86.76%	95.86%
Proposed method		Centroid cosine	55.25%	79.52%	87.38%
orthogonal transform	*Europarl*	Length normalized centroid cosine	55.38%	79.57%	87.69%
unit + dim-center	+	Closest cosine arithmetic mean	57.15%	82.75%	90.64%
global normalization	*Balanced news*	Closest cosine product (normalized)	60.02%	88.55%	96.37%
		Closest cosine product ($\epsilon = 0.1$)	60.23%	**89.43%**	96.34%
(Section 7)		Centroid cosine	55.22%	79.19%	87.38%
	Europarl	Length normalized centroid cosine	55.35%	79.48%	87.58%
	+	Closest cosine arithmetic mean	57.33%	82.65%	90.54%
	Full news	Closest cosine product (normalized)	60.17%	88.79%	**96.59%**
		Closest cosine product ($\epsilon = 0.1$)	**60.41%**	88.75%	96.29%

Table 18: Results on English-Spanish phrase translation selection. The best results for each method (denoted by blocks separated by double lines) are given in bold, and the best overall results are underlined.

Finally, it is important to note that, in spite of the promising results, it is the forward translation probability of the baseline phrase-based SMT system that obtains the best top 1, top 5 and top 10 accuracies. Nevertheless, we consider that these number should not be directly compared to the ones achieved by the different bilingual word embeddings because of two reasons. First of all, the forward translation probabilities are estimated from phrase level alignments, and this information is not available for the different bilingual word embedding methods. In this regard, it would be more appropriate to compare our results to the forward lexical weightings, which use the same word level information as we do. When doing so, we observe that the proposed method performs considerably better than the forward lexical weightings for the top 1 accuracy, with an absolute improvement of 2.96%. Even if the forward lexical weightings still get better top 5 and top 10 accuracies, these are closely followed by BiSkip. Apart from that, it should be taken into account that translation probabilities and lexical weightings are estimated in one direction (either forward or inverse), whereas the bilingual word embedding methods tested here, with the exception of Align-Init, do not work in any specific direction. This is a very important factor for the translation selection task under discussion, since an ideal system would choose the most probable translation candidate to get the highest possible accuracy, and not the most similar one. For instance, "loquacious" is as similar as "talkative", if not more, to the Spanish word "locuaz", yet less probable to be its translation, because it is a very uncommon word in English. This clearly favors the scores that are estimated in the relevant direction, as it is the case of the forward translation probabilities and the forward lexical weightings. However, when the full machine translation system is considered, the language model is able to compensate and even invert this advantage. In fact, the inverse translation probabilities and lexical weightings, despite their poor performance in the phrase translation selection task here, get a higher weight than their forward counterparts in an end-to-end phrase-based SMT system as shown in Table 15 for our experiments in Section 5.4. Considering these two aspects, we believe that our results are very promising and have the potential of being helpful in end-to-end machine translation, which we experimentally test next in Section 8.4.

8.4 Experiments on end-to-end machine translation

In this section, we test the different bilingual word embeddings and phrase similarity measures discussed throughout the chapter in an end-to-end English-Spanish machine translation task. For that purpose, we use the same experimental framework presented in Section 5.4, with the difference that, instead of dynamically adding the probability estimate of our logistic regression model into the phrase table, we statically add the phrase similarity score computed over the bilingual word embeddings in each case. The baseline system as well as the training, development and test sets are the same as the ones in Section 5.4, and we also perform 3 independent MERT runs for each configuration and report the average BLEU score for them.

Given the time necessary to run each experiments, we only tested what we considered the 3 most interesting configurations in Section 8.3 according to the experiments there.

			BLEU (%)	
			Development	**Test**
Baseline phrased-based SMT			**31.68**	28.28
Align-Init	Europarl	Length normalized centroid cos	31.65 (-0.03)	28.29 (+0.01)
BiSkip (word + context)	Europarl	Closest cos product (normalized)	31.63 (-0.05)	**28.32 (+0.04)**
Proposed method	Europarl + Full	Closest cos product ($\epsilon = 0.1$)	31.65 (-0.03)	28.28 (+0.00)

Table 19: Results on English-Spanish machine translation with phrase similarity scoring

Considering that Zou et al. (2013) report an improvement of 0.30 BLEU points on Chinese-English machine translation for their Align-Init method using the centroid cosine similarity, this was one of the configurations that we decided to test here. More concretely, we chose the word embeddings trained on Europarl and the normalized variant of the centroid cosine similarity, which give the best results for these settings in Section 8.3. In addition to that, we also tested the bilingual mapping method proposed in Section 7 trained on the full bilingual and monolingual corpora and using the closest word thresholded cosine product, which achieves the best top 1 accuracy among all the word embedding configurations tested in Section 8.3, as well as the BiSkip method taking the sum of word and context vectors trained on Europarl with the closest word normalized cosine product, whose top 1, top 5 and top 10 accuracies are all very close to those of the forward lexical weightings.

The obtained results are shown in Table 19. As it can be seen, in spite of the promising results in the phrase translation selection experiments in Section 8.3, none of the configurations tested is able to get any clear improvement over the baseline system. The case of Align-Init is particularly remarkable since, as mentioned before, the authors report an improvement of 0.30 BLEU points on Chinese-English machine translation under very similar conditions, while we only get an insignificant gain of 0.01 BLEU points. This suggests that the language pair could be one of the central reasons for our poor results, as distant languages like Chinese and English might hypothetically better benefit from distributional semantic information. Another factor to consider is that Zou et al. (2013) perform a single run of MERT, so the uncertainty around their BLEU score is higher than ours and their good results might, at least in part, be simply a product of chance.

In any case, it is remarkable that we are not able to get any significant improvement for the BiSkip and the proposed models either despite their superior results on the phrase translation selection experiments in Section 8.3. While we believe that these previous experiments depict a promising basis, we therefore conclude that there is some element that fails in the integration in an end-to-end SMT system. We believe that an error analysis in the phrase translation selection task could help better understand the behavior of the proposed model and identify the possible problem, which we leave as future work. In any case, when comparing our approach to other successful methods in the literature as seen in Section 4.4.1, we observe that the most relevant difference is that they use more sophisticated compositional models based on neural networks, which we consider a key point to explore in the future.

8.5 Conclusions

In this chapter, we have applied bilingual word embeddings for phrase translation similarity scoring. Instead of using complex neural network architectures to jointly learn the word embeddings and their compositionality and work with phrase embeddings, we have explored different metrics to directly compute a phrase similarity score over bilingual word embeddings. In this regard, our theoretical analysis of the centroid cosine similarity, a simple yet widely used metric for that purpose, shows that what it actually computes has little to do with how humans would assess phrase similarity based on word similarity, and we have proposed alternative phrase similarity measures that we believe better capture the human intuition.

For the purpose of intrinsically evaluating different bilingual embedding methods and phrase similarity measures, we have proposed a new task on phrase translation selection that relies solely on a phrase table from a standard phrase-based SMT system and a word aligned parallel test set, hence providing an inexpensive and robust evaluation framework that is closely related to machine translation. Our English-Spanish experiments show that one of the phrase similarity measures we propose, the closest word cosine product, achieves the best results in the task, beating the standard centroid cosine similarity measure by a large margin. Moreover, the bilingual mapping method proposed in Chapter 7 achieves the best top 1 accuracy among the different bilingual embedding models tested, with a considerable improvement over the lexical weightings used in SMT, which rely on the same word alignment information. It also proves to be the only method to clearly benefit from additional monolingual corpora, although it lags behind the BiSkip model by Luong et al. (2015) in the top 5 and top 10 accuracies.

In any case, in spite of the promising results in the phrase translation selection task, neither of these models nor the Align-Init method by Zou et al. (2013) yields to any clear improvement when integrated in an end-to-end English-Spanish SMT system. We believe that this could be partly attributed to our language pair, as Zou et al. (2013) report an improvement of 0.30 BLEU points on Chinese-English for their Align-Init method, while we only get an insignificant gain of 0.01 BLEU points applying the same method on English-Spanish under very similar conditions. We think that an error analysis in the phrase translation selection task, which we leave as future work, would be helpful to better understand the behavior of the proposed model and shed light on this lack of improvement in end-to-end machine translation.

9 Conclusions and future work

In this work, we have explored the use of distributional semantic and machine learning techniques to improve statistical machine translation. We have developed a logistic regression model for dynamic translation probability scoring and use it to incorporate additional lexical, source language phrase context and word clustering and word embedding based distributional semantic information into the translation model. In addition to that, we have developed a general framework to learn bilingual word embedding mappings and analyze it in relation to other existing methods both theoretically and experimentally. Finally, we have applied this and other existing bilingual word embedding models for phrase translation similarity scoring in SMT, for which we have analyzed the centroid cosine similarity and proposed alternative phrase similarity measures that address some of its issues.

Our experiments on English-Spanish machine translation show the effectiveness of the proposed logistic regression model with an improvement of up to 0.25 BLEU points over a strong baseline when using a basic set of lexical features. While it cannot be said that this improvement is big in quantitative terms, it is remarkably consistent across the different settings tested and relies solely on the bilingual corpus used to train the baseline SMT system itself. Our analysis of the weights assigned by MERT reveals that this improvement comes from almost completely replacing the forward translation probabilities while partly taking the place of the inverse translation probabilities and the language model, suggesting that the probability estimates given by our model are better than those used in standard phrase-based SMT and pointing to the usefulness of context features for lexical selection. This is in line with our theoretical expectations, as we prove our model to be a generalization of the standard translation probabilities used in phrase-based SMT that allows to incorporate additional features to obtain better probability estimates as shown by our experiments. We think that this provides a promising framework to incorporate additional information into the translation model, serving as a much more flexible alternative to factored models, even capable of dealing with dynamic context-dependent features.

Nonetheless, our results when incorporating distributional semantic information into the model through word cluster and word embedding features are not that positive. While we are able to improve the results obtained with the basic lexical features, in particular in the development set, this improvement is very modest and insufficient to draw any clear conclusion. In any case, our work in this regard was greatly conditioned by the high computational cost of running the experiments along with the strong hardware limitations that we had, and we plan to keep running more experiments for different feature sets and hyperparameters so as to obtain more conclusive results, in particular for the word embedding features.

Our work on bilingual word embedding mappings goes beyond machine translation and we believe makes important contributions in the field of bilingual word embeddings both from a theoretical and a practical perspective. The framework we propose starts with a basic optimization objective that is equivalent to the previous model by Mikolov et al. (2013b), and allows for several variants that we prove to be equivalent to other meaningful and relevant optimization objectives that are closely related to the previous methods by Xing et al. (2015) and Faruqui and Dyer (2014). This way, our framework provides a more global view of bilingual word embedding mappings, showing the underlying connection between the existing methods, revealing some flaws in their theoretical justification and providing an alternative theoretical interpretation for them. Our experiments on an existing English-Italian word translation induction and an English word analogy task give strong empirical evidence in favor of our theoretical reasoning, while showing that a variant of the proposed model clearly outperforms all the previous methods tested. More concretely, our model achieves the best bilingual performance in the word induction task, while preventing a degradation in monolingual quality and consequently performing at par with Xing et al. (2015) and considerably better than the rest of the methods in the monolingual word analogy task.

Finally, our work on the use of bilingual word embeddings for phrase translation similarity scoring explores a natural way to integrate this bilingual mapping method we propose as well as other existing bilingual word embedding models in a phrase-based SMT system. This requires generalizing the standard word level cosine similarity to a phrase level similarity measure, a compositional semantic problem that is also relevant for other tasks like document classification and information retrieval. In this regard, our theoretical analysis of the centroid cosine similarity, a simple yet widely used metric for that purpose, shows that what it actually computes has little to do with how humans would assess phrase similarity based on word similarity, and we propose alternative phrase similarity measures that we believe better capture the human intuition. For the purpose of intrinsically evaluating different bilingual embedding methods and phrase similarity measures, we propose a new task on phrase translation selection that relies solely on a phrase table from a standard phrase-based SMT system and a word aligned parallel test set, hence providing an inexpensive and robust evaluation framework that is closely related to machine translation. Our English-Spanish experiments show that one of the phrase similarity measures we propose, the closest word cosine product, achieves the best results in the task, beating the standard centroid cosine similarity measure by a large margin. Moreover, the bilingual mapping method we propose achieves the best top 1 accuracy among the different bilingual embedding models tested, with a considerable improvement over the lexical weightings used in SMT, which rely on the same word alignment information. It also proves to be the only method to clearly benefit from additional monolingual corpora, although it lags behind the BiSkip model by Luong et al. (2015) in the top 5 and top 10 accuracies. In any case, in spite of the promising results in the phrase translation selection task, neither of them nor the Align-Init method by Zou et al. (2013) yields to any clear improvement when integrated in an end-to-end English-Spanish SMT system. We believe that this could be partly attributed to our language pair, as Zou et al. (2013) report an improvement of 0.30 BLEU points on Chinese-English for their Align-Init method, while we only get an insignificant gain of 0.01 BLEU points applying the same method on English-Spanish under very similar conditions.

In the future, we would like to continue with this work while exploring new approaches to apply distributional semantic and machine learning techniques to improve machine translation. In particular, we plan to keep experimenting with different feature sets and hyperparameters for our logistic regression model for phrase translation probability scoring, in particular in relation to word embedding features. At the same time, we would like to explore the use of multiple logistic regression models for different configurations in the form of independent feature functions whose weights would be adjusted by MERT. Apart from that, we would like to perform an error analysis in the phrase translation selection task so as to better understand the behavior of our phrase translation similarity scoring model based on word embeddings and identify the possible reason for its poor performance in end-to-end English-Spanish machine translation. At the same time, we plan to explore the use of more sophisticated compositional models based on recurrent or recursive neural networks for building phrase embeddings from word embeddings and directly compute the phrase similarity score over them. Finally, we would like to go beyond the bilingual paradigm that is dominant in the field and focus on the use of multilingual word embeddings to exploit additional bilingual resources, which we believed can be particularly useful for less resourced languages.

References

Rodrigo Agerri, Josu Bermudez, and German Rigau. IXA pipeline: Efficient and ready to use multilingual NLP tools. In *LREC*, pages 3823–3828, 2014.

Eneko Agirre, Carmen Baneab, Claire Cardiec, Daniel Cerd, Mona Diabe, Aitor Gonzalez-Agirre, Weiwei Guof, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalceab, et al. Semeval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. 2015.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247, 2014.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 2003.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4): 467–479, 1992.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *The Journal of Machine Learning Research*, 11:2901–2934, 2010.

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics, 2005.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Alexander Clark. Combining distributional and morphological information for part of speech induction. In *Proceedings of the tenth conference on European chapter of the*

Association for Computational Linguistics-Volume 1, pages 59–66. Association for Computational Linguistics, 2003.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568*, 2014.

Susan T Dumais, George W Furnas, Thomas K Landauer, Scott Deerwester, and Richard Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285. ACM, 1988.

Chris Dyer, Victor Chahuneau, and Noah A Smith. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2013.

Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. Association for Computational Linguistics, 2014.

John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis*, 1957.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. Learning continuous phrase representations for translation modeling. In *ACL*, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016. URL `http://www.deeplearningbook.org`.

Stephan Gouws and Anders Søgaard. Simple task-specific bilingual word embeddings. In *Proceedings of NAACL-HLT*, pages 1386–1390, 2015.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. Bilbowa: Fast bilingual distributed representations without word alignments. *arXiv preprint arXiv:1410.2455*, 2014.

Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*, 13(1):307–361, 2012.

Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

Geoffrey E Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.

John Hutchins. Machine translation: A concise history. *Computer aided translation: Theory and practice*, 2007.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *ACL*, 2015.

Daniel Jurafsky and James H Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Pearson Prentice Hall, 2008.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. Inducing crosslingual distributed representations of words. 2012.

Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, 1999.

Tomáš Kočiskỳ, Karl Moritz Hermann, and Phil Blunsom. Learning bilingual word representations by marginalizing alignments. *arXiv preprint arXiv:1405.0947*, 2014.

Philipp Koehn. *Moses: User manual and code guide*, 2016.

Philipp Koehn and Hieu Hoang. Factored translation models. In *EMNLP-CoNLL*, pages 868–876, 2007.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 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, 2007.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Gorka Labaka. *EUSMT: incorporating linguistic information into SMT for a morphologically rich language. Its use in SMT-RBMT-EBMT hybridation.* PhD thesis, 2010.

Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861, 2014.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets.* Cambridge University Press, 2014.

Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. In *CoNLL*, pages 171–180, 2014a.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014b.

Peng Li, Yang Liu, and Maosong Sun. Recursive autoencoders for ITG-based translation. In *EMNLP*, pages 567–577, 2013.

Percy Liang. *Semi-supervised learning for natural language.* PhD thesis, 2005.

Wenpeng Lu and Ruojuan Xue. Comparative study on multi-systems combination in machine translation. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 8, pages 308–312. IEEE, 2010.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*, 2014.

Thang Luong, Hieu Pham, and Christopher D Manning. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, 2015.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing.* MIT press, 1999.

Sven Martin, Jörg Liermann, and Hermann Ney. Algorithms for bigram and trigram word clustering. *Speech communication*, 24(1):19–37, 1998.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780, 2006.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013b.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013c.

Marvin Minsky and Seymour Papert. *Perceptrons.* MIT press, 1969.

Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252, 2005.

Makoto Nagao. A framework of a mechanical translation between Japanese and English by analogy principle. *Artificial and human intelligence*, pages 351–354, 1984.

Franz Josef Och. 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, 2003.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51, 2003.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

DE Rumelhart, GE Hinton, and RJ Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011.

Harold Somers. An overview of EBMT. In *Recent advances in example-based machine translation*, pages 3–57. Springer, 2003.

Jinsong Su, Deyi Xiong, Biao Zhang, Yang Liu, Junfeng Yao, and Min Zhang. Bilingual correspondence recursive autoencoders for statistical machine translation. In *EMNLP*, 2015.

Simon Šuster. The Brown et al. 1992 clustering. Slides from the RUG Computational Linguistics reading group, 2013. URL `http://www.let.rug.nl/suster/talks/Brown_RUG.pdf`.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. Decoding with large-scale neural language models improves translation. In *EMNLP*, pages 1387–1392, 2013.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics, 1996.

Xing Wang, Deyi Xiong, and Min Zhang. Learning semantic representations for nonterminals in hierarchical phrase-based translation. In *EMNLP*, 2015.

Michael Wick, Pallika Kanani, and Adam Pocock. Minimally-constrained multilingual embeddings via artificial code-switching. 2015.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, 2015.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. Bilingually-constrained phrase embeddings for machine translation. In *ACL (1)*, pages 111–121, 2014.

Kai Zhao, Hany Hassan, and Michael Auli. Learning translation models from monolingual continuous representations. In *NAACL*, 2015.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398, 2013.