Unsupervised cross-lingual representation modeling for variable length phrases

Jingshu Liu

UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE NANTES

THÈSE DE DOCTORAT DE

L'UNIVERSITE DE NANTES
COMUE **U**NIVERSITE **B**RETAGNE **L**OIRE

Ecole Doctorale N°601
Mathèmatique et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

« Jingshu LIU »

« Unsupervised cross-lingual representation modeling for variable length phrases »

Thèse présentée et soutenue à NANTES , le 29 Janvier, 2020
Unité de recherche : LS2N
Thèse N° :

Rapporteurs avant soutenance :

Pierre Zweigenbaum, Directeur de recherche, CNRS - Université de Paris Saclay

Laurent Besacier, Professeur des universités, Université de Grenoble Alpes

Composition du jury :

Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition ne comprend que les membres présents

Président : Someone Someone

Examinateurs : Emmanuel Morin, Professeur des universités, Université de Nantes

Sebastián Peña Saldarriaga, Docteur en informatique, Easiware

Pierre Zweigenbaum, Directeur de recherche, CNRS - Université de Paris Saclay

Laurent Besacier, Professeur des universités, Université de Grenoble Alpes

Olivier Ferret, Ingénieur chercheur, CEA LIST

Dir. de thèse : Emmanuel Morin, Professeur des universités, Université de Nantes

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my supervisor Emmanuel Morin and my co-supervisor Sebastián Peña Saldarriaga for the continuous support of my Ph.D. study during these three years. I also appreciate their contributions of time, their patience, their support and inspiring advice which allow this Ph.D. bring innovative advance to both academic field and practical real-life application.

My sincere thanks go to Pierre Zweigenbaum, Laurent Besacier and Olivier Ferret for taking part of my thesis committee and for for their interest in my research, patient reading and insightful comments. I also thank them for their questions during the Ph.D. defense which incented me to widen my research from various perspectives.

I am more than grateful to all members of the TALN team and Dictanova with whom I have had the pleasure to work. It truly has been very good time in the lab and the company thanks to these outstanding and lovely people. A special thank to Joseph Lark for the effort for my paperwork and the basketball sessions after work.

Last but not the least, I would like to thank my family for all their unconditional love and endless encouragement. I also express my gratitude to all my friends for their presence, their help and their emotional support. Finally I thank my girlfriend Jia who always stood by my side through these years.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

As more and more text data of different languages are collected via millions of websites, it is in tremendous researchers' and enterprises' interests to leverage this data. Among various attractive applications (terminology extraction, opinion mining, machine translation, question answering, etc), bilingual phrase alignment is what this thesis studies. With recent advances in Machine Learning and Natural Language Processing, our work proposes a new unified bilingual phrase alignment framework in an unsupervised manner.

In the beginning section of this introductory chapter, we briefly discuss the two axes for bilingual phrase alignment which are the guidelines of this work. The second section presents the industrial context of our work. In the next section we give and clarify our phrase definition. Then from the fourth section to the sixth section we progressively describe several key features that help us achieve our final objective: unsupervised unified bilingual phrase alignment. Finally, the manuscript structure will be given at the end of the chapter.

1.1 Bilingual phrase alignment

Bilingual phrase alignment is an essential task for various NLP applications, ranging from phrase synonymy and phrase similarity to machine translation. Besides, projecting phrases into a common space can be an attractive feature when integrated in some industrial software. The objective consists in, given a phrase in source language, extracting the best translation phrase from the target language corpus.

The task requires two main technology axes: the phrase representation modeling and the bilingual phrase mapping with the properly learned representations.

The first axe is considered to be the prerequisite of the latter. For many years human has been trying to understand how the semantics of language sequence are captured. Traditionally, two complementary principles addressed this problem:

- **Compositional principle.** Compositionality is defined as the property where "the mean-

ing of the whole is a function of the meaning of the parts" (Keenan and Faltz, 1985). In Szabó (2017), the authors state that "The meaning of a complex expression is determined by its structure and the meanings of its constituents". For example, *a frying pan is indeed a pan used for frying* (Morin and Daille, 2012).

- **Syntactical principle.** The compositional principle would fail when encountering idiomatic expressions such as *a pain in the neck* (refers to something or somebody that is annoying or difficult to deal with according to Cambridge dictionary [1]) or *pomme de terre* (lit. apple of earth, meaning: potato) in French. The syntactical principle would like to complement the compositionality by analysing the sentence sequence which can be divided into clauses, and clauses can be further divided into phrases. Consequently these less compositional expressions are treated as a whole unit in a syntactical structure, e.g. a parsing tree.

For the first axis of this thesis, the phrase representation learning, we follow these two major principles as we desire to propose a unified framework to represent all types of phrases without length or linguistic constraints. Both compositional and idiomatic phrases are going to be handled in the same framework with respect to the compositionality and the syntactical structure.

The second axe is in between the *Information Retrieval* and the *Machine Translation* domains. While the task objective is to extract the most likely translation phrase by ranking all the candidates, it is not a task of generating the most likely translation sequence as in *Machine Translation*. Nonetheless, there are still a substantial amount of common points with *Machine Translation*. One might actually view the translation task as a sequential/conditional top 1 candidate selection process, in other words, the translator ranks all the candidate tokens and selects the best at each time step t with regard to all the previously generated tokens at time steps $[1, t-1]$. To sum up, the translation objective is to maximise the conditional probability:

$$\arg\max_{y} \prod_{t=1}^{T_y} P(y^t|x, y^1, y^2, ..., y^{t-1}) \tag{1.1}$$

where y^t means the selected best token at step t and x means the initial information. The factorial part in 1.1 can be also written as:

$$P(y^1, y^2, ..., y^t|x) = P(y^1|x)P(y^2|x, y^1), ..., P(y^t|x, y^1, y^2, ..., y^{t-1}) \tag{1.2}$$

[1] `https://dictionary.cambridge.org`

It is worth noting that there is one extreme case where the translation task is equivalent to the alignment task: when the target sequence length T equals to 1. Some call the word alignment task *bilingual word lexicon induction*. Therefore we estimate that the phrase alignment task is a non sequential translation task in terms of the candidate generative side and a sequence information retrieval task considering the comparison and selection aspect.

1.2 Industrial context

The thesis is carried out in an industrial context with the CIFRE[2] (*Convention Industrielle de Formation par la REcherche*, lit. Industrial Conventions Training by Research) convention. The CIFRE subsidizes any company that hires a PhD student to collaborate with a public laboratory. In our scenario, the collaboration was originally between the company Dictanova [3] and *Laboratoire de Sciences du Numérique de Nantes (LS2N)* [4].

Dictanova was a french start-up enterprise as a software editor in SaaS (Software as a Service) mode who provided semantic analysis for the *Customer Relationship Management* (CRM). Created in 2011 by several young researchers from the historical Nantes Atlantique Computer Science Laboratory (LINA) which later composed LS2N with other laboratories, the company received a financial aid from the National Competition of the newly founded innovative technology companies in 2012 and then in 2013 it was awarded by the Forum of the European Language Technologies Industry. The company was imbued with a strong academic scientific culture, particularly in the field of natural language processing. A use case of its software is illustrated in the figure below.

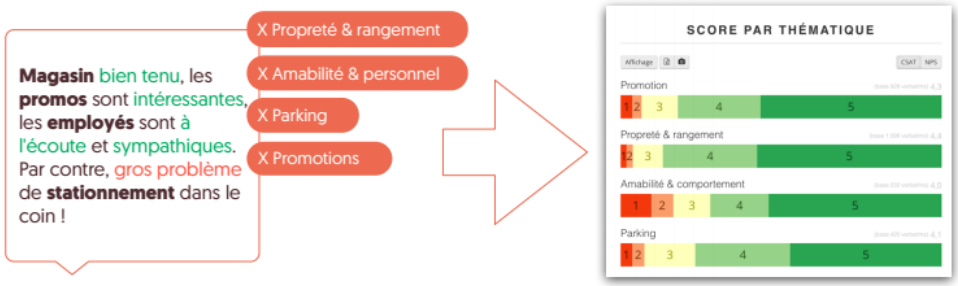


Figure 1.1 – A use case of the Dictanova semantic analysis.

[2] http://www.anrt.asso.fr/en/cifre-7843
[3] https://apps.dictanova.com/
[4] https://www.ls2n.fr/

The main function was to provide *aspect based sentiment analysis* [5] (ABSA), as shown in the Figure 1.1, the sentence in the left block is annotated with key phrases (bold), positive sentiment (green) and negative sentiment (red). Since the company supported several languages and the clients were also from different countries (France, Germany, Spain, etc), it was compelling for the company to develop a cross-lingual phrase alignment system. There are two major benefits:

- Facilitation of the transition from one language to another. Developing or maintaining a multilingual platform can be smoothed.
- Projection of the terms of different languages in a common space so that an international client has a better view of what is happening across all regions.

Moreover, recall the first axis of the thesis, the monolingual phrase representation modeling, which was also very interesting for the company as it enabled more subtle functions or manipulations over the phrases such as the automatic thematic clustering or synonym phrase aggregation.

In 2019, Dictanova was acquired by Easiware [6], who develops a multi-channel application processing software to enhance customer support service. As pointed out by the co-founder of Easiware, Brendan Natral, the perspective is to integrate the technology of Dictanova on the machine learning and the semantic analysis [7]. Concerning the thesis, since Easiware has also an international vision, our objective remains uninterrupted. We are able to continue our pre-acquisition work without major impact.

However, our industrial context also brings us some practical limits:

- The corpora are noisier compared to the academic ones and they are usually in very specialised domains. Besides, the size is often modest.
- Start-up enterprises are more product driven, and the less resource the system demands, the more practical it is for the enterprise to apply. With this perception, we would like to propose a system that requires as less calculation resources as possible.
- Syntactical information such as the parsing tree are not trivial information as not all the supported languages have a commercialised parser and in addition, the parsing process

[5] `https://en.wikipedia.org/wiki/Sentiment_analysis#Feature/aspect-based`
[6] `https://www.easiware.com/`
[7] `https://www.easiware.com/blog/easiware-acquisition-startup-dictanova`

could cause an excessive pre-processing burden for the relatively light product. Hence a system without requiring syntactical information becomes our leading choice.

- Parallel data are difficult to obtain for our specialised domain, and the cost of annotating parallel data is certainly expensive. However, we dispose naturally of comparable corpora as the clients speak usually of the same topics for a given customer. (*The delivery is too slow.* vs *Je suis déçu de la livraison. (lit. I am disappointed with the delivery.)*) Our final framework should be unsupervised or quasi unsupervised.

1.3 Phrase definition

Linguistically, the definition of a phrase is "is a group of words (or possibly a single word) that functions as a constituent in the syntax of a sentence, a single unit within a grammatical hierarchy"[8]. However, some argue that a phrase should contain at least two words (Finch, 2000). In our case, we apply the first definition where a phrase is simply a group of words or a single-word.

Sobin (2010) proposes to use tree structures to represent phrases, which provide schematics of how the words in a sentence are grouped and relate to each other. Any word combination that corresponds to a complete subtree can be seen as a phrase. This is in line with the compositional principle mentioned in 1.1.

Kroeger (2005) states that the meaning of a phrase is often determined by the syntactical head. For example, *blue* ***shoe***, *very* ***happy*** and ***watch*** *TV*. This suggests that we should take the syntactical structure into account when associating phrase components as discussed in the syntactical principle in 1.1.

1.4 Unified phrase representation

We would like to propose a unified phrase representation for phrases of all syntactical types of variable length. For instance, this may be observed by a few classes of semantically synonymous phrase pairs:

- **Length related**
  - **Single-word synonym pair**, as in *energy - power*.

[8] `https://en.wikipedia.org/wiki/Phrase`

  - **Same length multi-word synonym pair**, as in *to carry on - to keep going*.
  - **Fertile synonym pair**, as in *wind generator - aerogenerator*.

- **POS (part of speech) related**

  - **Same POS synonym pair**, as in *attitude - morale*.
  - **Different POS synonym pair**, as in *buy a car - purchase of a car*.

With these possible features taken into account, the unified phrase representation should be versatile and generic enough to allow that semantically close phrases are more similar to each other no matter how different they may appear on the superficial level, or close in the representation space, for instance the Euclidean vector space if we represent the phrases with vectors.

1.5 Bilingual unified phrase alignment

Once we have the unified monolingual phrase representation, the bilingual unified phrase alignment can be concluded by what has been mentioned in 1.1 plus the features in 1.4.

- **Length related**

  - **Single-word to single-word alignment (sw2sw)**, as in *bag - sac*.
  - **Same multi-word length alignment (n2n)**, as in *wind turbine - turbine éolien*.
  - **Fertile alignment (p2q)**, as in *airflow - flux d'air*.

- **POS (part of speech) related**

  - **Same POS alignment**, as in *car - voiture*.
  - **Different POS alignment**, as in *buy a car - achat d'une voiture*.

Because we do not have the POS information during the alignment most of the time, we focus more on the length criterion when analysing our systems. As for the POS related feature, we consider that it is automatically included and no further special processing is required for aligning different POS phrases.

1.6 Unsupervised alignment

Not only because of the limits mentioned in 1.2, but also the fact that even outside of the industrial context, parallel data is always costly to build as it requires specialised expertise. Besides such data is unfortunately often nonexistent for low-resource languages and specialised domains (Lample, Conneau, et al., 2018). Therefore it is preferred that the alignment system operates in an unsupervised manner.

More concretely, unsupervised means, in our scenario, that we do not have any access to cross-lingual information about phrases which in many cases is equal to a phrase mapping table. In our unsupervised proposal, we tackle this problem by incorporating pre-trained bilingual word embeddings (BWE) (Mikolov, Quoc V. Le, et al., 2013) and a back-translation mechanism (Sennrich et al., 2016a). The general idea is to exploit the shared vector space of the bilingual word embedding in a wider context where a sequence of words can also be projected into the same shared space.

1.7 Outline of the manuscript

This introductory chapter explains the motivation and the background of our work, which will be further detailed following the two axes (See 1.1) in a single-word and multi-word perspective.

From Chapter 2 to Chapter 3 we discuss the first axis concerning the phrase representation. Chapter 2 revises the traditional and neural network based approaches for word-level representation, in the most common case, it is in the form of word vectors (bag-of-words and word embeddings) in Euclidean vector space. We also review an effective method for improving the word vectors (bag-of-words and word embeddings) for corpora in specialised domain. Based on all these state-of-the art works, we propose our modifications which later improve our results on bilingual word alignment task. Chapter 3 generalises a substantial body of previous works on modeling multi-words. Finally, we propose a new structure for homogeneously modeling single-word and multi-word phrases, which fits better our scenario and allows a unified phrase representation.

Chapter 4 is focused on the training systems. From the regression based to the prediction based training system, we discover a new training strategy potentially powerful for many NLP tasks.

Then from Chapter 5 to Chapter 6, we dive into the second axis which seeks to align single-words or multi-words of different languages. Chapter 5 presents the word-level alignment. This

task is widely studied over the past years, we mainly cover the distributional based and the word embedding based approach as we mentioned in 2. Chapter 6 generalises the word-level alignment to multi-word level by incorporating a few techniques of the traditional compositional method and the Machine Translation field, reaching the best results for our task.

Chapter 7 concludes the thesis and opens perspectives for future works.

CHAPTER 2

WORD LEVEL REPRESENTATION

The compositionality of phrases necessitates meaningful word level representations in order to compose multi-word representations. In this chapter we first study word representation modeling with two widely applied approaches, the distributional approach and the distributed approach. Both represent words in n-dimensional Euclidean space $\mathbb{R}^n$. The first one is a more traditional approach where the word vectors are high dimensional sparse vectors with word co-occurrence, while the second one is a more recent approach where the word vectors are low dimensional dense vectors with learned parameters. Following the introduction of the two approaches, we explore how we can improve these approaches in our scenario where the domain-specific corpus size is quite modest. As these statistical approaches will often achieve better performance when larger training data is available, the mindset of the improvement is on how to efficiently add external data to reinforce the system. Then at the end of this chapter we explain an application of this improvement on dense word vectors.

2.1 Distributional representation

The distributional representation has been extensively studied in the NLP literature (Dagan et al., 1994; Lin, 1998; Kotlerman et al., 2010; P. D. Turney and Pantel, 2010; Baroni and Lenci, 2010) (and the references therein). This approach associates each word in a corpus vocabulary to a high dimensional (equal to the vocabulary size) vector space. A dimension in this space means a word-context co-occurrence, thus, naturally this kind of word vectors can be directly extracted from a word co-occurrence matrix. Suppose that each row of the matrix represents a word vector, then each column is a context to this particular word. We will give more details in the following section. Moreover, most works using the distributional approach apply an association measure to the word context vectors in order to smooth the vector values. In the second section 2.1.2, we review some most popular association measures and discuss their advantages and disadvantages.

2.1.1 Explicit vector space: word co-occurrence

The very first step towards the distributional approach is to construct a word co-occurrence matrix. For instance, each cell represents a word-context co-occurrence and the matrix is symmetric. To build such a matrix, we need to decide a window size within which we consider the proximity of words to be relevant. Typically this hyper parameter varies between one up to ten. The larger the window size is, the more features it will collect for the central word but also more likely to collect noisy features, while the smaller the window size is, the less noise it will retrieve but also the more discriminant information it will miss. For example, considering the following sentence:

> It is the first vehicle in the world in which passengers pay for their ride upon entering it.

With the vocabulary size d, the co-occurrence matrix will be in $\mathbb{N}^{d*d}$. Each word vector belongs to $\mathbb{N}^{d}$. The co-occurrence matrix corresponding to the sentence above will be in $\mathbb{N}^{16*16}$.

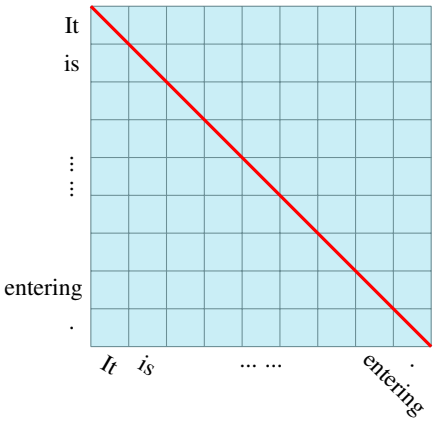


Figure 2.1 – Illustration of the co-occurrence matrix of the example sentence.

If the current central word is *passengers*, then within a window size of 3 and 5 we have following data where $v_i^{\text{passengers}}$ means the word vector of *passengers* at each dimension i.

window size = 3

> It is the first vehicle in the world in which passengers pay for their ride upon entering it.

$$v_i^{\text{passengers}} = \begin{cases} 1, & i \in \{\text{world, in, which, pay, for, their}\} \\ 0, & \text{otherwise} \end{cases}$$

window size = 5

It is the first vehicle in the world in which passengers pay for their ride upon entering it.

$$v_i^{\text{passengers}} = \begin{cases} 2, & i = \text{in} \\ 1, & i \in \{\text{the, world, which, pay, for, their, ride, upon}\} \\ 0, & \text{otherwise} \end{cases}$$

Note that within a window size of 5 we are able to associate the word *ride* to *passengers* which seems semantically relevant. Finally, the obtained word vectors have these properties:

1. **High-dimensional**. Obviously, as the co-occurrence matrix is in $\mathbb{R}^{d*d}$, each word vector has d dimensions. More concretely, if we look at some famous training corpora such as *Wiki*[1], *Gigaword*[2], *Common Crawl*[3] *or Europarl*[4] , the vocabulary size is usually between 100,000 to 500,000 even after some filtering.

2. **Sparse**. Most non functional words only co-occur with a limited number of other non functional words. Therefore the word vectors are highly sparse.

3. **Explicit**. The compelling point of the distributional representation is the explicitness of each dimension. One can simply understand the semantics behind each dimension and apply linguistic analysis.

2.1.2 Association measures

The raw word co-occurrence is often biased by the frequency of words in the training corpus, especially for the function words. Imagine that the word *in* appears 1000 times in an article and co-occurs 10 times with the word *passengers*, and the word *ride* appears only 5 times but

[1] `dumps.wikimedia.org/`
[2] `www.ldc.upenn.edu`
[3] `commoncrawl.org/`
[4] `www.statmt.org`

all these occurrences are within the window of *passengers*. From the pure word co-occurrence matrix, the word vector of *passengers* has a value of 100 at the dimension of *in* which is far more important than 5, the value at the dimension of *ride*. This phenomenon misleads us to a conclusion that the word *passengers* is semantically more related to *in* rather than *ride*. As a consequence, people introduce association measures to normalise the co-occurrence value with regard to the probability of having the relative word-context pair. The association measures studied are Pointwise Mutual Information (Fano, 1961), Log-likelihood (Dunning, 1993), and the Discounted Odds-Ratio (Evert, 2005).

Pointwise Mutual Information (PMI). Mutual Information reflects the mutual dependence between two random variables. For two discrete variables X and Y whose joint probability distribution is $P(x, y)$, and $P(x)$, $P(y)$ the marginal distributions, the mutual information between them, denoted $I(X, Y)$, is given by Shannon and Weaver (1949):

$$I(x, y) = \sum_x \sum_y P(x, y) \log \frac{P(x, y)}{P(x)P(y)} \tag{2.1}$$

The application of Pointwise Mutual information dates back to Fano (1961), he states that "if two points (words), x and y, have probabilities P(x) and P(y), then their mutual information, MI(x,y), is defined to be:"

$$PMI(x, y) = \log \frac{P(x, y)}{P(x)P(y)} \tag{2.2}$$

Many NLP tasks exploited this variant to construct word vectors (Church and Hanks, 1990; Dagan et al., 1994; P. Turney, 2001). Intuitively, $PMI(x, y)$ approaches $+\infty$ if there is a strong relation between x and y, and $-\infty$ if x and y are independent. In practice, word probabilities $P(x)$ and $P(y)$ are estimated by simply counting the number of observations of x and y in the training corpus and normalizing by N, the size of the corpus. Similarly, joint probabilities, $P(x, y)$, are estimated by counting how many times x co-occurs with y in the pre-set window, divided by the corpus size N. So if we take the previous example, let a be the occurrence of the word *passengers*:

$$\begin{aligned} PMI(\text{passengers, in}) &= \log \frac{\frac{10}{N}}{\frac{a}{N}\frac{1000}{N}} = \log \frac{N}{100a} \\ PMI(\text{passengers, ride}) &= \log \frac{\frac{5}{N}}{\frac{a}{N}\frac{5}{N}} = \log \frac{N}{a} \\ PMI(\text{passengers, ride}) &> (\text{passengers, in}) \end{aligned} \tag{2.3}$$

With PMI the value for the word context pair (*passengers, ride*) becomes higher than (*pas-*

sengers, in), which seems more logical compared to the raw co-occurrence. However, since the value of $\frac{P(x,y)}{P(x)P(y)}$ is almost always greater than 1, a problem with this measure is that the logarithm would overestimate low counts and underestimate high counts (Hazem and Morin, 2016).

Log Likelihood (LL). Likelihood function expresses how likely the parameters (θ) of a statistical model are while having a certain data (D), it is denoted by $L(\theta|D)$. In a real life scenario, we do not know the parameters θ, it is what we want to learn. If D is a set of discrete variables, then we have:

$$L(\theta|D) = P(D|\theta) \tag{2.4}$$

Since D is observable, we can estimate θ by maximizing the probability $P(D|\theta)$. Pertaining to our work, we consider that a word-context pair (x, y) is actually a data observation-parameter pair. In this way $P(x|y) = \frac{P(x,y)}{P(y)}$, finally the log likelihood is not very different from PMI:

$$\log L(y|x) = \log P(x|y) = \log \frac{P(x,y)}{P(y)} \tag{2.5}$$

Once again if we take our example with *passengers*, this time the value of $LL(passengers|in) = \log \frac{1}{100}$ and $LL(passengers|ride) = \log 1$, reaching the maximum in our scenario. Compared to PMI, LL's values range from negative to zero and it ignores the marginal occurrence of the central word. In addition, with the negative input of the logarithm function, the small counts would be quickly underestimated.

Discounted Odds-Ratio (DOR). The (logarithmic) *odds-ratio* can be interpreted by:

$$\text{odds-ratio}(x, y) = \log \frac{o_{11}o_{22}}{o_{12}o_{21}} \tag{2.6}$$

where o_{11} means the co-occurrence of the word x and the context y, o_{12} the times when y occurs without x, o_{12} the times when x occurs without y and o_{22} the total occurrence (words) without x and y:

$$\begin{aligned} o_{11} &= o(x, y) \\ o_{12} &= o(y) - o_{11} \\ o_{21} &= o(x) - o_{11} \\ o_{22} &= N - o_{12} - o_{21} - o_{11} \end{aligned} \tag{2.7}$$

As always, pertaining to the previous example, there is now a problem: the value of o_{12} is zero because all the five occurrences of *ride* co-occurs with *passengers*. This is also reported by

Evert (2005), indicating the odds-ratio "assumes an infinite value whenever any of the observed frequencies is zero ($-\infty$ for $o_{11} = 0$ or $o_{22} = 0$, $+\infty$ for $o_{12} = 0$ or $o_{21} = 0$)." Thus a lot of works apply a discounted version named *discounted odds-ratio* which avoids the infinitive value by adding a constant $\frac{1}{2}$:

$$\text{odds-ratio}_{disc}(x, y) = \log \frac{(o_{11} + \frac{1}{2})(o_{22} + \frac{1}{2})}{(o_{12} + \frac{1}{2})(o_{21} + \frac{1}{2})} \tag{2.8}$$

Actually we find that OR is nearly equivalent to PMI in pragmatical view:

$$\begin{aligned} \text{PMI} &= \log \frac{P(x, y)}{P(x)P(y)} \\ &= \log \frac{\frac{o(x,y)}{N}}{\frac{o(x)}{N}\frac{o(y)}{N}} = \log \frac{o(x, y)N}{o(x)o(y)} \end{aligned} \tag{2.9}$$

$$\begin{aligned} \text{odds-ratio} &= \log \frac{o_{11}o_{22}}{o_{12}o_{21}} \\ &= \log \frac{o(x, y)(N - (o(y) - o(x, y)) - (o(x) - o(x, y)) - o(x, y))}{(o(y) - o(x, y))(o(x) - o(x, y))} \\ &= \log \frac{o(x, y)N + o(x, y)(o(x, y) - o(x) - o(y))}{o(x)o(y) + o(x, y)(o(x, y) - o(x) - o(y))} \\ &\text{let } o(x, y)(o(x, y) - o(x) - o(y)) \text{ be } c\text{:} \\ &= \log \frac{o(x, y)N + c}{o(x)o(y) + c} \end{aligned} \tag{2.10}$$

Note that DOR only applies a smoothing on OR so the proportion does not change for non zero values. We can see that the final forms in 2.9 and 2.10 are quite close, normally the absolute value of c is much smaller than $o(x, y)N$ and $o(x)o(y)$, therefore DOR and PMI tend to be close in most cases. Our analysis coincides with the comparison experiments between DOR and PMI on the bilingual word alignment task conducted by Hazem and Morin (2016).

The results in Figure 2.2 show that DOR and PMI have extremely similar performance which confirms our intuition. Considering most works exploit PMI and it is theoretically more steady than LL, we choose to use PMI in our frameworks as the association measure of the distributional approach. After the association process, the co-occurrence matrix becomes a real matrix $\mathbb{R}^{d*d}$.

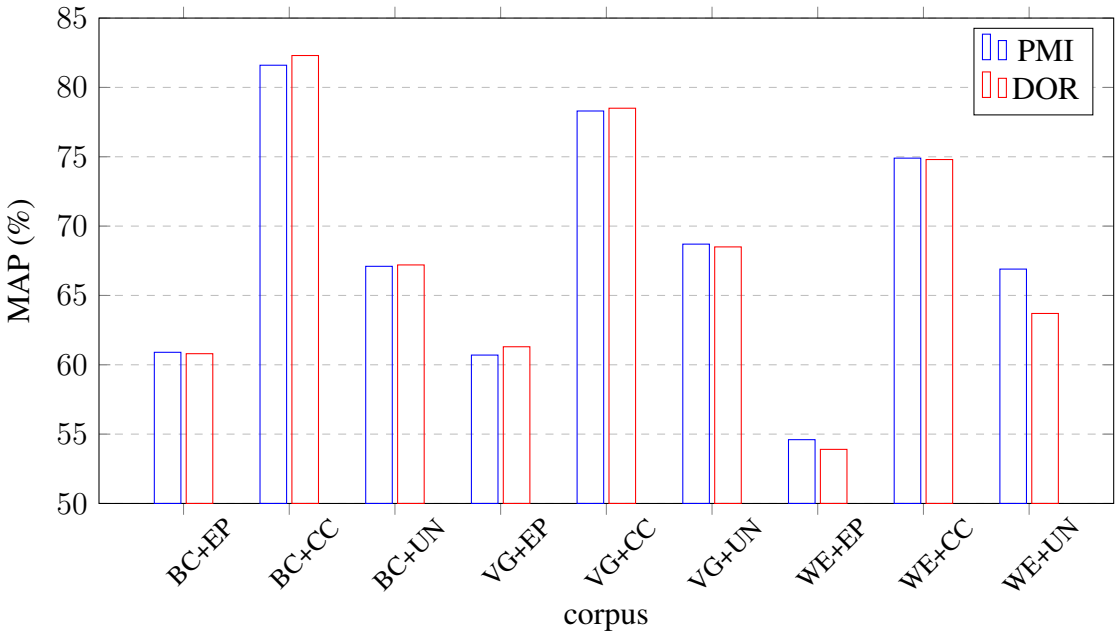


Figure 2.2 – Comparison between DOR and PMI on experiments of Hazem and Morin (2016). BC (breast cancer), VG (volcanology) and WE (wind energy) are corpora in specialized domain with roughly 500k, 400k and 300k tokens. Europarl (EP), common crawl (CC) and united nations (UN) are general domain corpora with roughly 60M, 85M and 380M tokens. For corpus merging method, we will further explain it in Section 2.3.

2.2 Distributed representation

Despite the simpleness and the effectiveness of the explicit word vectors, it can always be tremendously time and space consuming to train a model, especially when the corpus size becomes as big as the *wiki* corpus which is a rather common one for training many NLP models. Another way of constructing word vectors is by using neural network based approaches, the generated word vectors are low-dimensional, dense, and the dimensions features are highly generalised. We call this kind of word vectors *word embeddings*. In this section we first cover the basic component of these approaches: the Neural Network. Then we will explain the two most widely exploited systems: CBOW and Skip-gram (Mikolov, Sutskever, et al., 2013; Mikolov, K. Chen, et al., 2013), and quickly discuss some other popular word vectors like *Glove* (Pennington et al., 2014) and *fastText* (Bojanowski et al., 2017).

2.2.1 Neural Network: sparse to dense vector space

McCulloch and Pitts (1943) proposed to simulate the behaviour of human neurons using a mathematical model, opening an era of artificial neural network research. However it was not until the great progress of the machine computing power that the artificial neural network came to dominate a wide range of machine learning tasks. Thanks to the parallel processing ability of the GPUs, neural networks or deep neural networks are successfully addressed in reasonable time. A neural network can take an input vector of any dimension, so sparse high-dimensional vectors can be transformed to dense low-dimensional vectors.

The basic component of a neural work, like in neuroscience, consists in a **neuron**. A neuron can be considered as a perceptron unit (Rosenblatt, 1958) (in other words, a perceptron is a single neuron). A neuron is called activated or excited when the received information passes the threshold:

$$\begin{aligned} z &= w^T x + b \\ a &= f(z) \end{aligned} \tag{2.11}$$

$x \in \mathbb{R}^n$ is the input vector, w is a weight matrix in $\mathbb{R}^{n*d}$ with d the output dimension. f is a sigmoid function (S-shaped curve) in order to imitate the human neuron. We call it **Activation Function** in artifical neural network models. The most popular sigmoid functions are logistic (σ), $tanh$, and *rectified linear unit* ($relu$) function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.12}$$

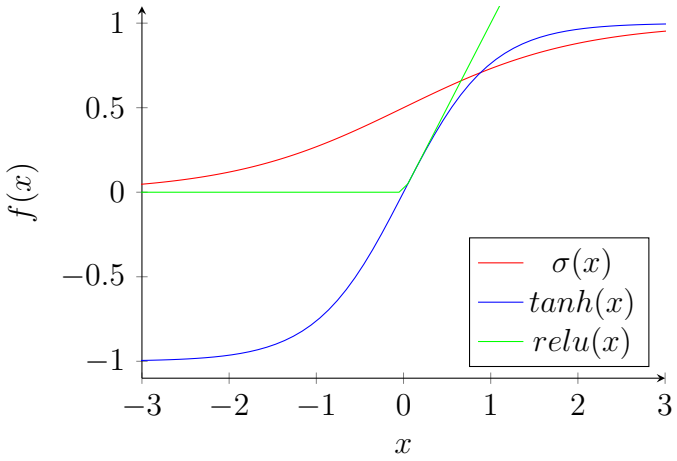


Figure 2.3 – Comparison of three different activation functions.

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.13}$$

$$relu(x) = \max(0, x) \tag{2.14}$$

We can plot these functions in a single space to compare them in Figure 2.3. The $tanh$ function is actually a zoomed and translated version of σ as $tanh(x) = 2\sigma(2x) - 1$. The *rectified linear unit* is supposed to be more similar to human neurons as it creates sparse representations with true zeros, which seem remarkably suitable for naturally sparse data (Glorot et al., 2011).

A deep neural network is usually composed of multiple layers where each layer takes the output of the last layer and generates a new output. Eventually, each layer is connected by sequentially passing the information. The most common neural network layer is the **Fully-connected** layer. A multi-layer fully connected neural network has shown impressive performance in transforming information into a more generalized feature space.

As shown in Figure 2.4, a neural network with L fully-connected layers has following key properties:

- n^l. The dimension size at layer l.
- f_l. The activation function at layer l.
- $W^l \in \mathbb{R}^{n^l * n^{l-1}}$. The weight matrix for transforming the information from the layer $l-1$ to l.

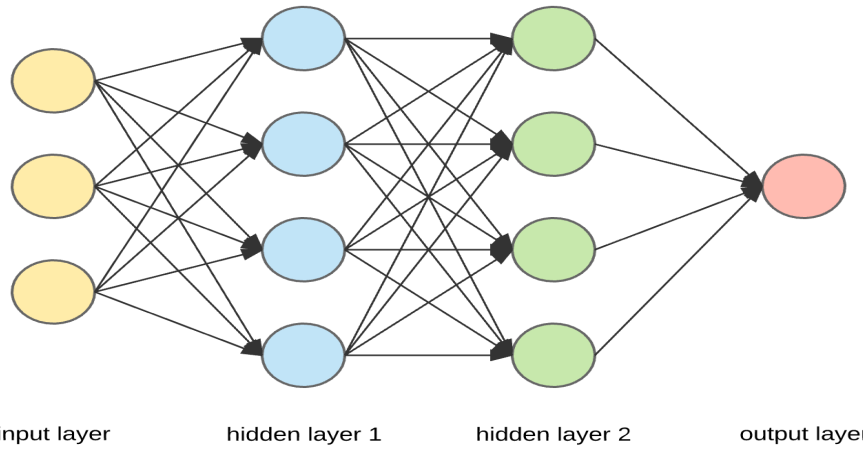


Figure 2.4 – An illustration of a neural network with multiple fully connected layers.

- $b^l \in \mathbb{R}^{n^l}$. The bias for the l-th layer.
- $z^l \in \mathbb{R}^{n^l}$. The state for the neurons in the l-th layer.
- $a^l \in \mathbb{R}^{n^l}$. The activation for the neurons in the l-th layer.

The network transfers information layer by layer following the equation 2.11. Final output y equals to a^L. With a training set sample $(x^{(i)}, y^{(i)}), 1 \leqslant i \leqslant N$, the objective function of the network is:

$$J(W, b) = \sum_{i=1}^{N} J(W, b; x^{(i)}, y^{(i)}) \tag{2.15}$$

Our objective is to minimize $J(W, b; x^{(i)}, y^{(i)})$, with **gradient descent**, we can update the network's parameters:

$$\begin{aligned} W^l &= W^l - \alpha \sum_{i=1}^{N} (\frac{\partial J(W, b; x^{(i)}, y^{(i)})}{\partial W^l}) \\ b^l &= b^l - \alpha \sum_{i=1}^{N} (\frac{\partial J(W, b; x^{(i)}, y^{(i)})}{\partial b^l}) \end{aligned} \tag{2.16}$$

where α is the learning rate. Note that W and b are all the weight matrices and bias vectors in each layer which can also be denoted by $(W^1, W^2, ..., W^L)$ and $(b^1, b^2, ..., b^L)$. The question consists in how we calculate the partial differentials. For instance, by the **chain rule**, the partial differential for the weight matrix W can be written as:

$$\frac{\partial J(W, b; x^{(i)}, y^{(i)})}{\partial W_{i,j}^l} = (\frac{\partial J(W, b; x^{(i)}, y^{(i)})}{\partial z^l})^T \frac{\partial z^l}{\partial W_{i,j}^l} \tag{2.17}$$

For the layer l, we let $\delta^l = \frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial z^l} \in \mathbb{R}^{n^l}$ to represent the partial differential of the final output about z^l. In addition, since $z^l = W^l a^{l-1} + b^l$, we have:

$$\frac{\partial z^l}{\partial W^l_{i,j}} = \frac{\partial (W^l a^{l-1} + b^l)}{\partial W^l_{i,j}} = \begin{pmatrix} 0 \\ . \\ . \\ a^{l-1}_j \\ . \\ . \\ 0 \end{pmatrix} = a^{l-1}_j \tag{2.18}$$

The equation 2.17 can then be written as:

$$\begin{aligned} \frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial W^l_{i,j}} &= (\delta^l)^T \frac{\partial z^l}{\partial W^l_{i,j}} \\ &= \begin{pmatrix} \delta^l_0 & . & . & \delta^l_i & . & . & \delta^l_{n^l} \end{pmatrix} \begin{pmatrix} 0 \\ . \\ . \\ a^{l-1}_j \\ . \\ . \\ 0 \end{pmatrix} = \boxed{\delta^l_i a^{l-1}_j} \end{aligned} \tag{2.19}$$

Therefore we have:

$$\frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial W^l} = \delta^l (a^{l-1})^T \tag{2.20}$$

In the same way:

$$\frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial b^l} = \delta^l \tag{2.21}$$

Now let's look at the calculation of δ^l:

$$\begin{aligned} \delta^l &= \frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial z^l} \\ &= \frac{\partial a^l}{\partial z^l} \quad \frac{\partial z^{l+1}}{\partial a^l} \quad \frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial z^{l+1}} \end{aligned} \tag{2.22}$$

For each of the three parts, we have:

$$\frac{\partial a^l}{\partial z^l} = \frac{\partial f_l(z^l)}{\partial z^l} \tag{2.23}$$
$$= \boxed{\textbf{diag}(f_l'(z^l))} \quad \text{Because every activation } f_l \text{ is an element-wise function.}$$

$$\frac{\partial z^{l+1}}{\partial a^l} = \frac{\partial W^{l+1}a^l + b^{l+1}}{\partial a^l} = \boxed{(W^{l+1})^T} \tag{2.24}$$

$$\frac{\partial J(W, b; x^{(i)}, y^{(i)})}{\partial z^{l+1}} = \boxed{\delta^{l+1}} \tag{2.25}$$

With 2.23, 2.24 and 2.25, the equation 2.22 is equivalent to :

$$\begin{aligned} \delta^l &= \frac{\partial J(W, b; x^{(i)}, y^{(i)})}{\partial z^l} \\ &= f_l'(z^l) \odot (W^{l+1})^T \delta^{l+1} \end{aligned} \tag{2.26}$$

We can see that δ of the layer l can be calculated by the δ of the next layer $l+1$. It can be considered as a product of the gradient of the current activation function and the weighted δ of the next layer using the weight matrix of the next layer. Therefore in order to calculate the δ of layer l, we have to begin with obtaining the δ of the previous layer. Recursively, we should start from the very last layer. This procedure is basically the famous **Back propagation** algorithm:

Sometimes the **Back propagation** algorithm will have a *gradient vanishing* problem with sigmoid activation functions when there are a lot of layers. Since the differential is always inferior to 1, after several layers the product by a series of value below 1 would become ignorable. One possible solution is to use the *relu* as the activation function.

In implementation, it can take a tremendous time to calculate all the differentials in a network graph. Most machine learning libraries use **auto-differentiation** to solve the gradients [5]. The idea is to convert all functions into several derivative-known mathematical operations such as $+, -, \times, /, \log, \exp, cos, sin, x^a$, etc. And then apply the **chain rule** on the computational graph where each node represents a basic mathematical operation and each leaf an input vari-

[5] Theano, Tensorflow, Chainer, PyTorch and Deeplearning4j (beta version). Note that Theano and Tensorflow use static computational graph which is highly parallelizable but cannot be changed after compilation, while Chainer and Pytorch use dynamic computational graph which is more space consuming but highly flexible.

Algorithm 1: Back propagation algorithm.

Data: $(x^{(i)}, y^{(i)}), 1 \leqslant i \leqslant N$ (N training samples)
Result: W, b (the network parameters)
Random initialization of W and b ;
while *current epoch* $\leqslant$ *max epoch* ***or*** *early stop condition not met* **do**
  for $i = 1, ..., N$ **do**
    Forward pass of the entire network to get all z and a of each layer;
    Backward pass step 1 : calculate δ for every layer using 2.26;
    Backward pass step 2 : calculate the differentials for W and b of each layer :

$$\frac{\partial J(W, b; x^{(i)}, y^{(i)})}{\partial W^l} = \delta^l (a^{l-1})^T$$

$$\frac{\partial J(W, b; x^{(i)}, y^{(i)})}{\partial b^l} = \delta^l$$

    Update W and b of each layer with gradient descent:

$$W^l = W^l - \alpha \sum_{i=1}^{N} (\frac{\partial J(W, b; x^{(i)}, y^{(i)})}{\partial W^l})$$

$$b^l = b^l - \alpha \sum_{i=1}^{N} (\frac{\partial J(W, b; x^{(i)}, y^{(i)})}{\partial b^l})$$

  end
end

able or tensor. The final differential can be obtained by simply multiplying the known derivative of each operation. In case of multiple paths, we sum all the derivatives.

Based on the derivative calculation order, we can do the auto-differentiation in a **forward mode** or **backward mode**. In the forward mode, the derivative is calculated in the same order as in the computation graph, while in the backward mode, the order is reversed by first calculating the derivative of the last operation. The backward mode is essentially equivalent to **back propagation**.

2.2.2 CBOW

The most successful neural network based word vectors were introduced by Mikolov, Sutskever, et al. (2013). Two models were presented: *Continuous Bag-of-Words* (CBOW) and *Skip-gram*. We first discuss the CBOW model in this section.

The *Continuous Bag-of-Words* Model is composed of two fully connected neural network layers without bias ("biases are not used in the neural network, as no significant improvement of performance was observed") and one softmax layer at the end in order to obtain a probability distribution on the vocabulary with the size of V. The input are **one hot vectors** $x \in \mathbb{N}^V$. Representing a word by a one hot vector means that the value at only the dimension of the word equals to 1, at other dimensions the values are always 0. A word w is represented by x:

$$x_i, i \in [0, V] = \begin{cases} 1 & i \text{ is the index of } w. \\ 0 & \text{otherwise} \end{cases} \tag{2.27}$$

Let's begin with the simplest case where we consider only one context word w_I, we note the first layer the hidden layer and the second layer the output layer. The weight matrix in the hidden layer is $W_1 \in \mathbb{R}^{V*N}$ and the weight matrix in the output layer is $W_2 \in \mathbb{R}^{N*V}$. The input one hot vector x of the input word w_I is transformed to the hidden dense vector h by:

$$h = W_1^T x = W_1^T[k, :] = v_{w_I}^T \tag{2.28}$$

Since x is a one hot vector, the output vector v_{w_I} is actually one row of the matrix W_1. Then from the hidden layer to the output layer, each word in the vocabulary is associated with a score

which is stored in the vector u, and each value u_j can be obtained by a scalar product:

$$\begin{aligned} u &= W_2^T h \\ u_j &= W_2[:, j]^T h = s_{w_j}^T h \end{aligned} \tag{2.29}$$

s_{w_j} is the j-th column of the matrix W_2. Finally, a softmax function is applied to the output vector to get a probability distribution:

$$P(w_j|w_I) = y_j = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{2.30}$$

The probability of predicting the word w_j given the input word w_I is also denoted by y_j. According to 2.28 and 2.29, 2.30 can be written as:

$$P(w_j|w_I) = y_j = \frac{exp(s_{w_j}^T v_{w_I}^T)}{\sum_{j'=1}^{V} exp(s_{w_j'}^T v_{w_I}^T)} \tag{2.31}$$

The **training objective** is to maximize the conditional probability of observing the actual output word w_O (let $\hat{j}$ be the corresponding index in the vocabulary) given the input context word w_I with regard to the network parameters (weight matrices).

$$\begin{aligned} \arg\max_{W_1,W_2} P(w_O|w_I) &= \arg\max_{W_1,W_2} y_{\hat{j}} \\ &= \arg\max_{W_1,W_2} \log y_{\hat{j}} \\ &= \arg\max_{W_1,W_2} u_{\hat{j}} - \log \sum_{j'=1}^{V} exp(u_{j'}) \\ &= \arg\min_{W_1,W_2} \log \sum_{j'=1}^{V} exp(u_{j'}) - u_{\hat{j}} \\ E &= \log \sum_{j'=1}^{V} exp(u_{j'}) - u_{\hat{j}} \end{aligned} \tag{2.32}$$

Finally the objective is equivalent to minimize E, in order to update W_1 and W_2, we use the back propagation explained in 1. First we should calculate the derivative of E with regard to

the output vector where u_j is the j-th element.

$$\begin{aligned}
\frac{\partial E}{\partial u_j} &= \frac{\partial \log \sum_{j'=1}^{V} exp(u_{j'}) - u_{\hat{j}}}{\partial u_j} \\
&\text{if } j \neq \hat{j}, \text{ by chain rule :} \\
&= \frac{\partial \log \sum_{j'=1}^{V} exp(u_{j'})}{\partial \sum_{j'=1}^{V} exp(u_{j'})} \frac{\partial \sum_{j'=1}^{V} exp(u_{j'})}{\partial exp(u_j)} \frac{\partial exp(u_j)}{\partial u_j} - 0 \\
&= \frac{1}{\sum_{j'=1}^{V} exp(u_{j'})} 1 \, exp(u_j) \\
&= y_j \\
&\text{similarly, if } j = \hat{j} : \\
&\frac{\partial E}{\partial u_j} = y_j - 1
\end{aligned} \tag{2.33}$$

Next we calculate the derivative of the loss on W_2 to obtain the gradient for W_2 whose function is to transform the hidden vectors to the network's output layer:

$$\begin{aligned}
\frac{\partial E}{\partial W_2[i,j]} &= \frac{\partial E}{\partial u_j} \frac{\partial u_j}{\partial W_2[i,j]} \\
&= \begin{cases} (y_j - 1)h_i & \text{if } j = \hat{j} \\ y_j h_i & \text{otherwise} \end{cases} \quad = e_j h_i (e_j \text{ denotes the } \mathbb{1} \text{ part})
\end{aligned} \tag{2.34}$$

So the **update function** for W_2 is:

$$\begin{aligned}
W_2^{t+1}[i,j] &= W_2^t[i,j] - \eta e_j h_i \\
&\text{Or :} \\
s_{w_j}^{t+1} &= s_{w_j}^t - \eta e_j h
\end{aligned} \tag{2.35}$$

where t means the t-th iteration and η the learning rate. s_{w_j} is the j-th column of the matrix W_2.

As for the W_1, the gradient is calculated by the derivative of the loss E on W_1:

$$\begin{aligned}
\frac{\partial E}{\partial W_1[k,i]} &= \frac{\partial E}{\partial h_i} \frac{\partial h_i}{\partial W_1[k,i]} \\
&= \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \frac{\partial u_j}{\partial h_i} \frac{\partial \sum_{k=1}^{V} W_1[k,i] x_k}{\partial W_1[k,i]} \\
&= \sum_{j=1}^{V} e_j W_2[i,j] \; x_k \\
&= EH_i x_k \text{ (EH is an N-dim vector denoting } \sum_{j=1}^{V} e_j W_2[i,j])
\end{aligned} \tag{2.36}$$

Note that Equation 2.36 follows Equation 2.26 and Equation 2.20, the first part is essentially the δ and the second part is the layer input. Vector EH can be intuitively interpreted as the weight sum of output vectors of all words in vocabulary with regard to their prediction error e_j. Finally the **gradient for the weight matrix** W_1 in the hidden layer can be obtained from Equation 2.36:

$$\frac{\partial E}{\partial W_1} = xEH^T \in \mathbb{R}^{V*N} \tag{2.37}$$

The update function for W_1 is:

$$\begin{aligned}
W_1^{t+1} &= W_1^t - \eta x EH^T \\
&\text{Or :} \\
v_{w_I}^{t+1} &= v_{w_I}^t - \eta EH^T
\end{aligned} \tag{2.38}$$

where v_{w_I} is one row in W_1, the row index is determined by the non-zero element index in x as x is a one-hot vector. So basically for each update of W_1, there is only one row that is updated.

The real input of CBOW model is built on **multiple context words**, this is not very different from what we have seen. Instead of taking one single input vector x, the multi-word context model takes the average of C input vectors $[x^1, x^2, ...x^C]$, so the only change point is the hidden

vector h:

$$\begin{aligned} h &= W_1^T \frac{\sum_{c=1}^{C} x^c}{C} \\ &= \frac{\sum_{c=1}^{C} v_{w_c}^T}{C} \end{aligned} \tag{2.39}$$

The loss function the same as in 2.32, and the update function for the weight matrix in the output layer (W_2) is also the same as in 2.35. The update function for the weight matrix in the hidden layer (W_1) is:

$$\begin{aligned} & W_1^{t+1} = W_1^t - \eta \frac{\sum_{c=1}^{C} x^c}{C} EH^T \\ & \text{Or :} \\ & v_{w_{I,c}}^{t+1} = v_{w_{I,c}}^t - \frac{1}{C}\eta EH^T, \text{ for } c \in [1, C] \end{aligned} \tag{2.40}$$

where x^c is one input context word one-hot vector, $v_{w_{I,c}}$ is one row of W_1 of the c-th word in the input context. This time for each update there will be C updated rows.

2.2.3 Skip-gram

Skip-gram (Mikolov, Sutskever, et al., 2013) is the reversed architecture of CBOW. In place of taking several context words and predicting the central word, the models takes the central word as input and predicts its C context words. As a consequence the input to hidden transformation is the same as in Equation 2.28.

On the output layer, instead of outputting one single vector representing the multinomial distribution, the Skip-gram model outputs C same vectors:

$$\begin{aligned} u_c &= W_2^T h \\ u_{c,j} &= u_j = W_2[:, j]^T h = s_{w_j}^T h, c \in [1, C] \end{aligned} \tag{2.41}$$

where u_c is one output vector and $u_{c,j}$ is the j-th value of u_c. s_{w_j} is the j-th column of the hidden-to-output matrix W_2, s_{w_j} also represents the word w_j in the vocabulary before the softmax process. Next with the softmax function we obtain a probability distribution:

$$P(w_{c,j}|w_I) = y_{c,j} = \frac{exp(u_{c,j})}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{2.42}$$

The **training objective** is yet again very close to what we have seen in Equation 2.32, this time we want to maximize the probability of predicting all the actual c output context words

$w_{O,1}, w_{O,2}, ..., w_{O,C}$ given the input central word w_I :

$$\begin{aligned}
\arg\max_{W_1,W_2} P(w_{O,1}, w_{O,2}, ..., w_{O,C}|w_I) &= \arg\min_{W_1,W_2} -\log P(w_{O,1}, w_{O,2}, ..., w_{O,C}|w_I) \\
&= \arg\min_{W_1,W_2} -\log \prod_{c=1}^{C} \frac{exp(u_{c,\hat{j}_c})}{\sum_{j'=1}^{V} exp(u_{j'})} \\
&= \arg\min_{W_1,W_2} C \sum_{j'=1}^{V} \log exp(u_{j'}) - \sum_{c=1}^{C} u_{c,\hat{j}_c} \\
&= \arg\min_{W_1,W_2} C \sum_{j'=1}^{V} \log exp(u_{j'}) - \sum_{c=1}^{C} u_{\hat{j}_c} \\
E &= C \sum_{j'=1}^{V} \log exp(u_{j'}) - \sum_{c=1}^{C} u_{\hat{j}_c}
\end{aligned} \tag{2.43}$$

where $\hat{j}_c$ is the index of the actual c-th context word in the vocabulary. With the loss function E, we can calculate the derivative of E on every output vector $u_{c,j}$ following the same logic as in Equation 2.33:

$$\frac{\partial E}{\partial u_{c,j}} = y_{c,j} - \mathbb{1}(j) = e_{c,j} \tag{2.44}$$

where $\mathbb{1} = 1$ when $j = \hat{j}_c$. We define a V-dimensional vector EI as the sum of prediction errors over all actual context words:

$$EI_j = \sum_{c=1}^{C} e_{c,j} \ , j \in [1, V] \tag{2.45}$$

The derivative of the loss E with regard to W_2 can be then obtained:

$$\begin{aligned}
\frac{\partial E}{\partial W_2[i,j]} &= \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \frac{\partial u_{c,j}}{\partial W_2[i,j]} \\
&= \sum_{c=1}^{C} e_{c,j} h_i \\
&= EI_j h_i
\end{aligned} \tag{2.46}$$

Then the **update function** for W_2 is:

$$\begin{aligned}
W_2^{t+1}[i,j] &= W_2^t[i,j] - \eta EI_j h_i \\
&\text{Or :} \\
S_{w_j}^{t+1} &= S_{w_j}^t - \eta EI_j h
\end{aligned} \tag{2.47}$$

The derivative of the loss E with regard to W_1 is the same as in Equation 2.36, only this time EH is the sum of all the c output vectors' prediction errors or in other words, e_j in Equation 2.36 is replaced by EI_j :

$$\begin{aligned} \frac{\partial E}{\partial W_1} &= xEH^T \in \mathbb{R}^{V*N} \\ EH_i &= \sum_{j=1}^{V} EI_j W_2[i,j],\ , i \in [1,N] \end{aligned} \tag{2.48}$$

Finally the **update function** for W_1 is the same as in Equation 2.38.

2.2.4 Other popular word embeddings

There are a bunch of other word embeddings (Deerwester et al., 1990; Lebret and Collobert, 2014; Mnih and Kavukcuoglu, 2013; Mikolov, K. Chen, et al., 2013; Mikolov, Sutskever, et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017) which have been widely applied as the input of many NLP frameworks. In this section we will review three notable models because they are the more relevant to our work. The first one is rather a variant or an improvement of the work of Mikolov, Sutskever, et al. (2013), *Negative Sampling*, proposed by Mikolov, Sutskever, et al. (2013) themselves. The second one is *GloVe* (Pennington et al., 2014) which is a min least squares model with global learning (vs on-line learning in CBOW and Skip-gram). Finally the third model, *FastText*[6], is essentially a skip-N-gram model where each word is represented as a bag of character n-grams.

CBOW and Skip-gram with Negative Sampling

The negative sampling was originally proposed to optimize the computation for both CBOW and Skip-gram. By observing Equation 2.35 and 2.47, we can tell that updating the hidden-to-ouput weight matrix W_2 requires iterating through every word w_j in the vocabulary and running the whole network with them. One possible solution is to reduce the number of words to go through.

The actual output word (i.e., positive sample) should not be dropped, so the column in W_2 for the positive sample is always going to be updated. In addition, a few negative samples (the authors state that selecting 5-20 words works well for smaller datasets, and 2-5 words for large datasets) are randomly selected in order to update other weights in W_2. The selection could

[6] `https://fasttext.cc/`

be any probabilistic distribution by empirical tuning. The authors have exploited a unigram distribution raised to the $\frac{3}{4}$ power for the word counts, where more frequent words are more likely to be selected as negative samples.

$$P(w_i) = \frac{\#(w_i)^{\frac{3}{4}}}{\sum_{j=1}^{V} \#(w_j)^{\frac{3}{4}}} \tag{2.49}$$

where $\#(w_i)$ means the count of the word w_i in the training corpus. If we look at Figure 2.5, the intent of this distribution is to decrease the probability of more frequent words (e.g., grammatical words) compared to lexical words.

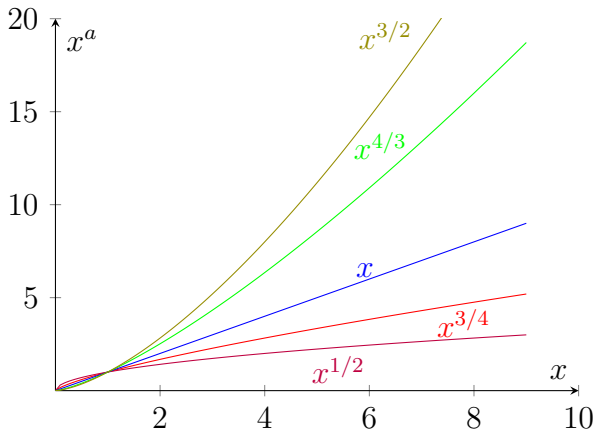


Figure 2.5 – Comparison of five different power functions.

With the set of negative samples $\mathcal{W}_{neg}$, the authors argue that a simplified loss function enables the model to produce high-quality embeddings, which is further elaborated in Goldberg and O. Levy (2014):

$$E = -\log \sigma(s_{w_O}^T h) - \sum_{w_k \in \mathcal{W}_{neg}} \log \sigma(-s_{w_k}^T h) \tag{2.50}$$

where s_{w_O} is the column in W_2 corresponding to the actual output word (positive sample) w_O and h is the output of the hidden layer which is $\frac{1}{C}\sum_{c=1} C v_{w_c}$ in the CBOW model and v_{w_I} in the Skip-gram model.

We first want to obtain the update equation for W_2 which is also s_{w_j} for $w_j \in w_O \cup \mathcal{W}_{neg}$.

The derivative of E with regard to s_{w_j} is :

$$\begin{aligned}\frac{\partial E}{\partial s_{w_j}} &= \frac{\partial E}{\partial s_{w_j}^T h}\frac{\partial s_{w_j}^T h}{s_{w_j}} \\ &= (\sigma(s_{w_j}^T h) - \mathbb{1}(w_j))h \\ \mathbb{1}(w_j) &= \begin{cases} 1 \text{ , if } w_j = w_O \\ 0 \text{ , if } w_j \in \mathcal{W}_{neg} \end{cases}\end{aligned} \tag{2.51}$$

Hence the **update function** for W_2 is :

$$s_{w_j}^{t+1} = s_{w_j}^t - \eta(\sigma(s_{w_j}^T h) - \mathbb{1}(w_j))h \tag{2.52}$$

This new update function makes the model consider only a subset of the entire vocabulary, $w_j \in w_O \cup \mathcal{W}_{neg}$, with a much smaller size of $1 + \#(\mathcal{W}_{neg})$ compared to V. This equation can be used for both CBOW and the Skip-gram model. For the skip-gram model, we apply this equation for one actual context word (w_O) at a time.

Next to obtain the update function for W_1 we need to calculate the derivative of E with regard to h :

$$\begin{aligned}\frac{\partial E}{\partial h} &= \sum_{w_j \in w_O \cup \mathcal{W}_{neg}} \frac{\partial E}{\partial s_{w_j}^T h}\frac{\partial s_{w_j}^T h}{h} \\ &= \sum_{w_j \in w_O \cup \mathcal{W}_{neg}} (\sigma(s_{w_j}^T h) - \mathbb{1}(w_j))s_{w_j}\end{aligned} \tag{2.53}$$

In Skip-gram :

$$= \sum_{w_O \in \mathcal{W}_{pos}} \sum_{w_j \in w_O \cup \mathcal{W}_{neg}} (\sigma(s_{w_j}^T h) - \mathbb{1}(w_j))s_{w_j}$$

This equation is EH in Equation 2.36 and 2.48. Finally for the **update function** for W_1 in CBOW, we simply replace EH with this new equation, and as for Skip-gram, we calculate the sum of this equation for each actual context word (positive sample) w_O and replace EH with it.

Since the authors of Mikolov, Sutskever, et al. (2013) declare that Skip-gram with negative sampling has empirically better results, in our work we exploit primarily Skip-gram with negative sampling as our word embedding vectors.

GloVe: Global vectors

Unlike CBOW and Skip-gram, Pennington et al. (2014) argue that these "shallow window-based methods suffer from the disadvantage that they do not operate directly on the co-occurrence statistics of the corpus. " They propose to learn word vectors based on the co-ocurrence of words of the entire training corpus.

The first step consists in constructing a word co-occurrence matrix X. Let $\sum_k X[i,k]$ denote the number of times any word appears in the context of word i. The training is achieved by addressing a weighted least squares regression problem with a weight function f:

$$E = \sum_{i,j}^{V} f(X[i,j])(w_i^T \tilde{w}_j + b_i + \tilde{b_j} - \log(X[i,j]))^2 \tag{2.54}$$

where w_i is a word vector for word i and $\tilde{w_j}$ is a word vector for a word in the context window of word i. b_i and $\tilde{b_j}$ are scalar bias for the corresponding word. And the weight function f is manually defined by the authors:

$$f(x) = \begin{cases} (x/x_{max})^\alpha & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{2.55}$$

where the authors have empirically set x_{max} to 100 for their huge training corpus (>6B tokens), and a power α of $\frac{3}{4}$ is reported to obtain the best results, which is an interesting coincidence with the power in negative sampling technique of Mikolov, Sutskever, et al., 2013. The objective of this function is to assign a relatively small weight for frequent co-occurrences and a linear weight for co-occurrences that are below the threshold x_{max}, as shown in the figure below.

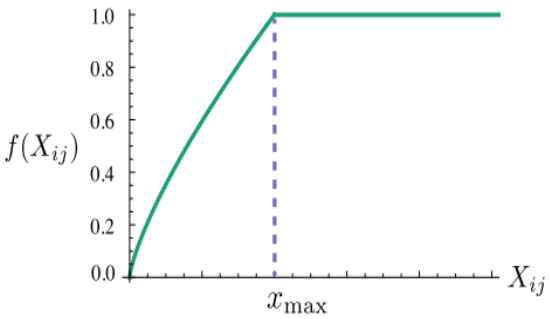


Figure 2.6 – Weight function in GloVe.

The gradients are fairly easy to calculate, for instance for w_i:

$$\begin{aligned}
&\frac{\mathrm{d}E}{\mathrm{d}w_i^T} = \frac{\partial f(X[i,j])(w_i^T \tilde{w_j} + b_i + \tilde{b_j} - \log(X[i,j]))^2}{\partial w_i^T} \\
&\text{let } y = (w_i^T \tilde{w_j} + b_i + \tilde{b_j} - \log(X[i,j]))^2 : \\
&= \frac{\partial f(X[i,j])y^2}{\partial y^2} \frac{\partial y^2}{\partial y} \frac{\partial y}{\partial w_i^T} \\
&= f(X[i,j]) \cdot 2y \cdot w_j^T
\end{aligned} \tag{2.56}$$

Following the same logic, we have:

$$\frac{\mathrm{d}E}{\mathrm{d}w_j} = f(X[i,j]) \cdot 2y \cdot w_i \tag{2.57}$$

$$\frac{\mathrm{d}E}{\mathrm{d}b_i} = \frac{\mathrm{d}E}{\mathrm{d}b_j} = f(X[i,j]) \cdot 2y \tag{2.58}$$

Pertaining to our work, we are particularly inspired by the weight function introduced in GloVe as it is closely related to what we want achieve by penalizing frequent co-occurrences, we will discuss the details later.

FastText: word embeddings with subword information

Bojanowski et al. (2017) propose to incorporate character-level information into word embeddings. Character-level or subword n-gram information is not original in word representation learning, Lazaridou, Marelli, et al. (2013), Botha and Blunsom (2014), Qiu et al. (2014), and Wieting et al. (2016) exploit these morphological information with supervised training setting (they have paraphrase pairs), while Kim et al. (2016), Sennrich et al. (2016b), and Luong and Christopher D. Manning (2016) model language with subword information with only corpus texts.

The authors of *FastText* build their model based on Skip-gram with negative sampling, they state that the Skip-gram model ignores the internal structure of words using a distinct vector representation for each word. They represent each word as a bag of character n-gram. They also include the whole word sequence in the n-gram list and two special n-grams at the beginning and end of words (*begin* = <, *end* = >). For example, the word "where" will have the following tri-gram representation:

<wh, whe, her, ere, re>, where

In practice, they extract all the n-grams for n greater or equal to 3 and smaller or equal to 6. In the vector space, for instance, we have a vocabulary of all n-grams $\mathcal{G}$, the vocabulary of n-grams for a given word w is denoted by $\mathcal{G}_w$. The word vector of w is the sum of the n-gram vectors in $\mathcal{G}_w$. Therefore, the scalar product of $s_{w_O}^T h$ in 2.50 becomes:

$$s_{w_O}^T h = \sum_{g \in \mathcal{G}_{w_O}} s_g^T h \tag{2.59}$$

In our experiments, our input word embedding vectors are primarily obtained by FastText as it is has shown the state-of-the art performance compared to Skip-gram with negative sampling on word similarity and analogy tasks.

2.3 Data selection for modest corpora with distributional representation

All the word vector models mentioned above will work very well on large corpus in open domain. Nevertheless, in our real life scenario, most of our corpora have a modest size on specialised domains. This would yield an unreliability of word co-occurrences. Considering that the statistics of word co-occurrences in a corpus is the primary source of information available to all unsupervised methods for learning word representations, both the distributional and distributed approach may possibly deteriorate with these unreliable counts. In this section, we present a popular solution to improve the reliability of word co-occurrences counts for building a distributional model.

The basic idea of the solution is to add data from other corpora in general domain because such corpora are easier to get, which has already been successfully employed in *machine translation* (MT) especially with *statistical machine translation* (SMT) (Moore and Lewis, 2010; Axelrod et al., 2011; Longyue Wang et al., 2014). This approach is also known as *data selection*, which improves the quality of the language and translation models, and hence, increases the performance of SMT systems. Hazem and Morin (2016) propose to apply this approach to bilingual lexicon extraction from specialized comparable corpora which is closely related to our work as the corpora we dispose are naturally specialized comparable corpora as discussed in 1.2. Their hypothesis is that word co-occurrences learned from a general-domain corpus for general words (as opposed to the terms of the domain) improve the characterization of the specific vocabulary of the corpus (the terms of the domain).

In the work of Hazem and Morin (2016), two data selection techniques have been proposed to enhance the performance of bilingual lexicon extraction:

The first one is called *Global Standard approach* which merges the whole external data into the domain-specific corpus and then constructs the word co-occurrence matrix. It is worth mentioning that by using this technique, we potentially increase quadratically the word co-occurrence matrix in memory space. For instance, we note V_s and V_g the vocabulary size for the domain-specific and open-domain corpora respectively. The size of word co-occurrence matrix is illustrated in Figure 2.7.

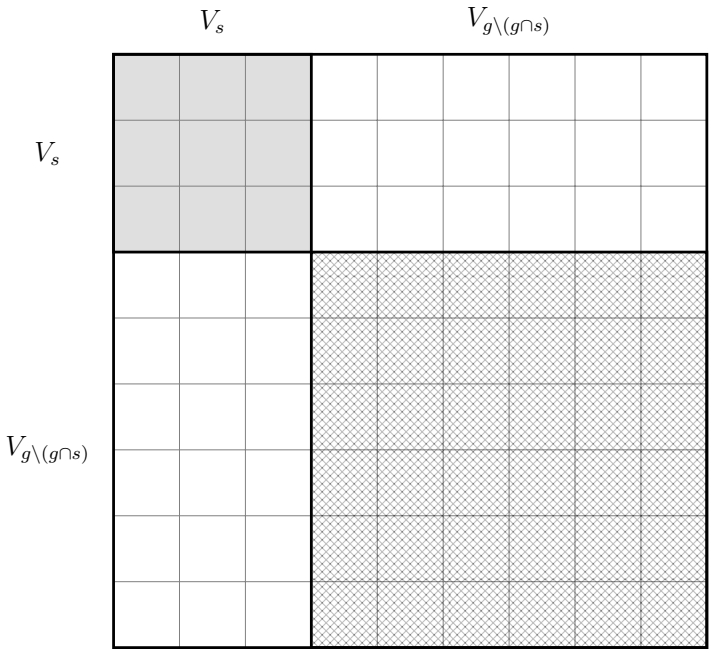


Figure 2.7 – GSA co-occurrence matrix.

where $V_{g\setminus g\cap s}$ means the vocabulary size of all the words that appear in the open-domain corpus but not in the specialized-domain corpus. The original word co-occurrence matrix is built exclusively on the specialized-domain corpus, its size is $V_s * V_s$. The merged matrix based on the merged corpus has a size of $(V_{g\setminus g\cap s} + V_s) * (V_{g\setminus g\cap s} + V_s)$, the number of elements stored in the matrix is boosted to the square of the increase of the vocabulary. Moreover, normally, $V_{g\setminus g\cap s} >> V_s$, the matrix obtained by *GSA* is far more space consuming than the original matrix.

The second approach named *Selective Standard Approach*, is more space efficient, since it enlarges only linearly the word co-occurrence matrix. The first step is to build independently the word co-occurrence matrix of the two corpora (specialized and general). The second step

is composed of merging co-occurrence counts for each word that appears in the two corpora, which allows to filter open domain words that are not part of the specialized corpus and renders the selective standard approach much less time consuming than the global standard approach. The process can be formally concluded as:

$$\forall w \in S \cap G, \forall c \in S \cup G,\ cooc(w,c) = cooc_S(w,c) + cooc_G(w,c) \tag{2.60}$$

where S and G represents the specialized and open-domain corpus. $cooc(w,c)$ is the word-context co-occurrence between the word w and the context c. An example of the diagram of this approach is shown in Figure 2.8.

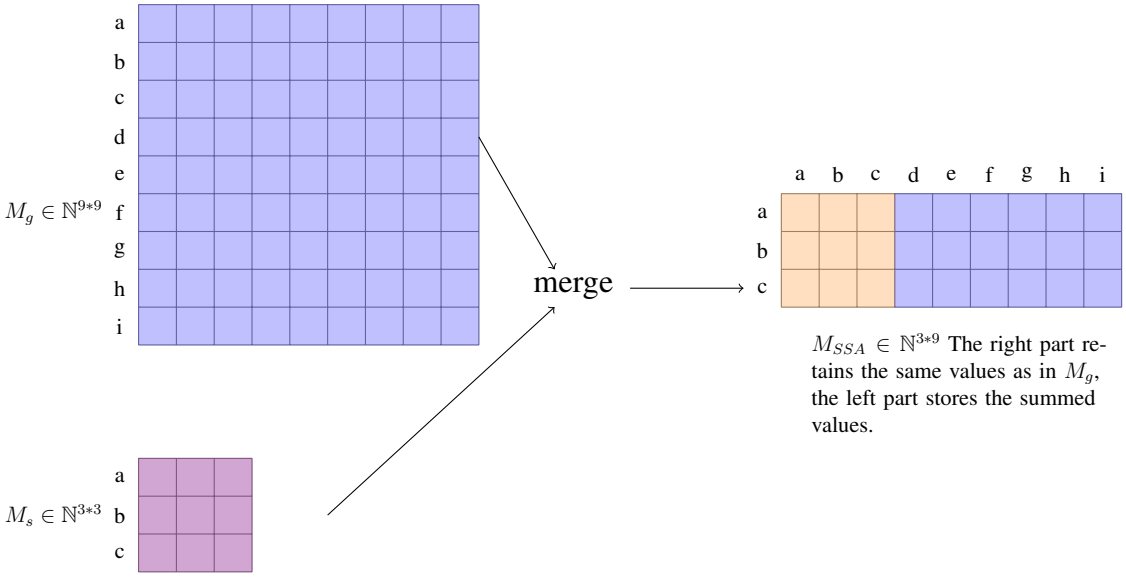


Figure 2.8 – SSA merging process.

Note that the merging process is carried out before applying the association measure mentioned in 2.1.2. The final matrix is of size $3 * 9$, compared to $9 * 9$ which will be the size using *GSA*. In this case, *SSA* is 6 ($V_{g \setminus g \cap s}$) times more efficient. Since $V_{g \setminus g \cap s}$ is generally much greater than V_s, this approach could save potentially a huge amount of memory.

The authors evaluate the *GSA* and *SSA* approaches on bilingual lexicon extraction task. We report their results in Figure 2.9.

First, the results confirm the effectiveness of the *data selection* approaches as they obtain significant improvements. In addition, since *GSA* and *SSA* have very similar performances, and in many cases *SSA* shows stronger results, we decide to apply the *SSA* merging technique when we want to exploit external data to enhance our data taking the space complexity into account.

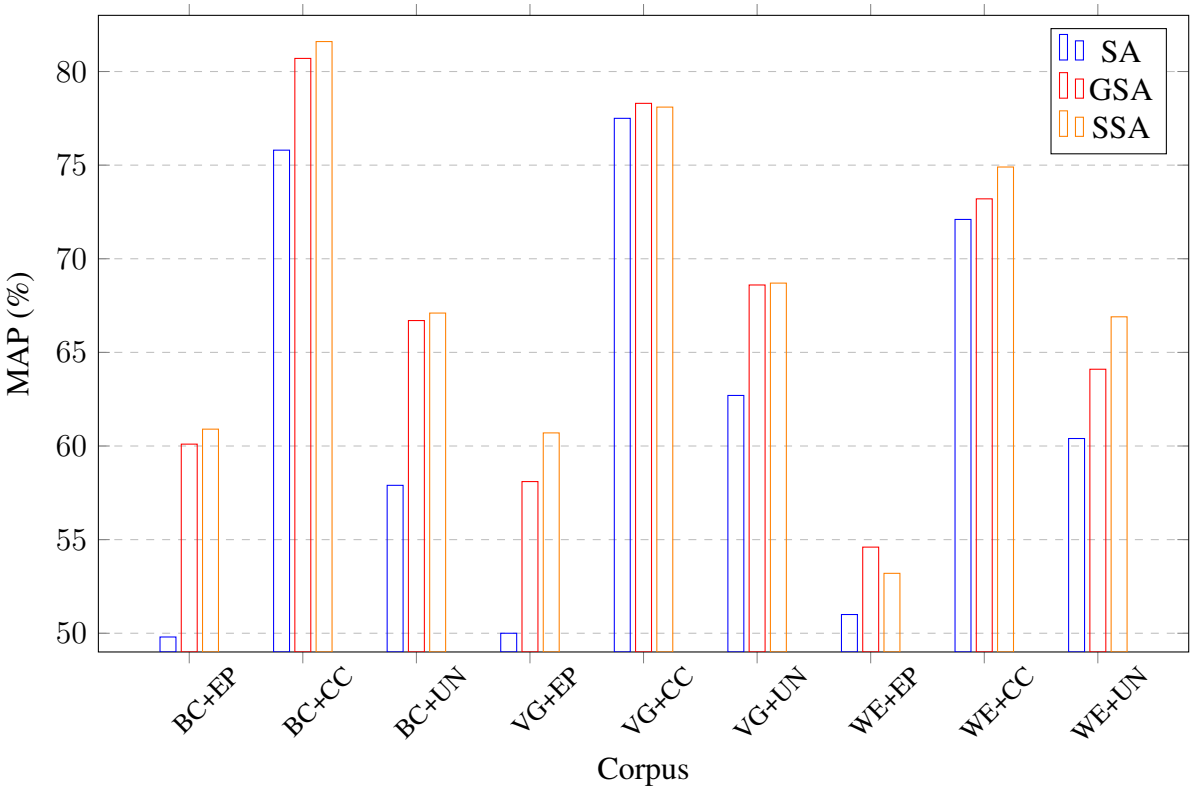


Figure 2.9 – Comparison between only external data (SA), Global data merging (GSA) and selective data merging (SSA) approach on bilingual lexicon extraction task with distributional word vectors using PMI as association measure. The x-axis labels represent the training corpora, the left left part represents the specialized-domain corpus and the right part the open-domain one. In case of the only external data setting (SA), only the open-domain corpus is used for constructing distributional word vectors. BC (breast cancer), VG (volcanology) and WE (wind energy) are corpora in specialized domain with roughly 500k, 400k and 300k tokens. Europarl (EP), common crawl (CC) and united nations (UN) are general domain corpora with roughly 60M, 85M and 380M tokens.

2.4 Contribution: Data selection with distributed representation

We have seen an efficient solution for training word co-occurrence count based word vectors. One would imagine how external data could be leveraged for distributed word vectors, or rather, word embeddings. In this section, we present our proposal of leveraging external data for distributed word vectors.

First, some of the distributed word vectors can directly apply the existing *data selection* approaches mentioned in the previous section. For example, *GloVe* can be easily constructed from a merged word co-occurrence matrix. While on-line learning models such as CBOW or Skip-gram which are built on iterating the entire corpus, the overall word co-occurrence do not play any role in the training. Our proposal concerns essentially these models, the objective is to reinforce the word embeddings trained on specialized domain corpora with external open-domain data.

Word embedding systems usually need a large amount of data in order to obtain reliable word vectors. However the size of domain-specialized comparable corpora is generally very modest (fewer than one million words for our real life corpora). The word embedding models trained on our small size specialized corpora are not capable of generalizing meaningful features. To overcome this problem, the idea exploited by Hazem and Morin (2016) using external open-domain data to enrich the training phase could be useful, but it occasionally makes the specialized word representation biased by the open-domain corpus. This is especially the case when the specialized corpus contains some infrequent or ambiguous words that have different meanings in different corpora. Considering these factors, we propose to use a concatenated vector of the one trained on the specialized corpora and on the open-domain corpora as our new word vector. More specifically, we want to concatenate a relatively small size word vector from the specialized corpora and a relatively large size one from the open-domain corpora. The concatenated vector will have a dimension size of $dim_s + dim_g$, where dim_s and dim_g represent the dimension size of the word embedding vectors trained on the specialized and general domain corpus. In our experiments, the word vector size for the word embedding model trained on the specialized corpus is empirically set to be 100 and the size for the model trained on the general corpus is set to be 300. Hence we preserve the specialized corpus features albeit with less corresponding weight features in the transformation matrix. Consequently the new concatenated word vector carries self-contained information from both corpora. Another advantage is that by doing the concatenation, it is not necessary to retrain our word embedding models over large

corpora because we can use existing pre-trained models available. This could save a great deal of time in practice while improving efficiency.

We evaluate our approach on the word synonymy task which is closely related to our real-life demands. To this end, we first train our own Skip-gram word embedding model on the specialized corpora we dispose of (WE-en, WE-fr, BC-en) [7]. The negative sampling size is set to 15. After 20 epochs we obtain our 100-dimensional word embeddings carrying specific domain-related information. Then we concatenate these vectors to the pre-trained 300-dimensional *FastText* embeddings, forming our final word embeddings of 400 dimensions. The results are shown in Table:

Dataset	**word embeddings**		
	S	G	$S \frown G$
WE-fr	4.61	4.95	**5.48**
WE-en	9.02	8.43	**22.93**
BC-en	23.63	25.02	**34.03**

Table 2.1 – Word synonynmy results (MAP%) on three datasets of two different languages. S and G mean that the word embeddings are seperately trained from the specialized and the general domain corpus. $S \frown G$ means that word embeddings are obtained from concatenating the seperately trained embeddings.

Across all the results, the concatenated embeddings reach the best performances. In parallel, when comparing the results of the first two columns, we find that it is not always guaranteed that the embeddings trained on the specialized domain corpus or on the open domain corpus leads to the better performance. Because it depends on the specificity of the words in the test list, for example, *rotor* is more domain related than *machine* in most situations, it may be more difficult for the open-domain model to find a closer relation for these two words. However, when concatenating the two models, we mitigate this problem by "pulling" the domain-related words together. Additionally, the vectors trained on a much larger corpus will carry relatively reliable statistics to our concatenated vectors, making them more resilient to the bias due to the small size of the specialized domain corpus size.

[7] Details of our data can be found in Appendix A.1.1 and A.1.2

2.5 Synthesis

In this chapter, we started by reviewing the word co-occurrence count based approach, also known as *distributional approach* for generating word vector representations. In summary, the vectors obtained from this approach are **high-dimensional**, **sparse** and **explicit**.

The integer word vectors are then normalized by an association measure which turns the co-occurrence matrix into a real matrix (the word vectors are thus real vectors), and meanwhile adjust the co-occurrence values with respect to the total occurrences of the central and the context word. We have presented three popular association measures: **Pointwise Mutual Information (PMI)**, **Log Likelihood (LL)** and **Discounted Odds-Ratio (DOR)**. In our work, we choose to use PMI because it is more stable than LL and theoretically similar to DOR but more efficient.

Then we have covered the **distributed approaches** which, on the contrary to the distributional approach, generate **low-dimensional**, **generalized** and **dense** word vectors, also known as *word embeddings*. We have particulary focused on the neural network based word embeddings because nowadays they hold most of the state-of-the-art results.

After revising the prerequisites for learning the parameters of a neural network, the most widely exploited word embeddings, **CBOW** and **Skip-gram** are explained so that we clearly understand how these embeddings are learned and what each hyper-parameter means exactly. This will be very useful when we want to train and fine-tune our own models.

Apart from the CBOW and Skip-gram, we have also included several other word embedding models which are interestingly related to our work. For instance, we would like to incorporate the **negative sampling** in order to accelerate our distributed model training, the weight function of **GloVe** to improve the performance of the distributional approach and **FastText** vectors to take sub-word information into account.

The final part of this chapter addresses the problem of generating relevant word vectors from modest corpora. We first introduced the **data selection** proposal which seeks external data to help training the model. Then at last we propose a new approach of data selection for word embeddings which has shown strong empirical results in our monolingual word synonymy task.

CHAPTER 3

MULTI-WORD LEVEL REPRESENTATION : PHRASE MODELING

Having studied the word level representation, we now move on to the multi-word level representation. Recall that the definition of *phrase* in Section 1.3 is no more than a group of words (or one single-word), which is basically equivalent to *multi-word*. Learning multi-word representations is a key step towards our final goal. The single-word representation discussed in the previous chapter plays actually an important role for building multi-word representations because most multi-word frameworks either use single-word tokens as input or even more simply, treat multi-words as single-word tokens. This chapter begins with approaches that do not depend on learnable parameters. Thanks to their simplicity and fast applicability, these approaches are often considered as the baseline approaches. Thereafter, we investigate the approaches that are based on the same architecture as in the word embedding work of Mikolov, Sutskever, et al., 2013. In the next section, we review some neural networks which can encode a sequential input. Note that all these networks use the same back-propagation algorithm to calculate the gradients for updating the whole network (See 2.2.1). The following section introduces some recent models for sequence modeling. These models have become the latest state-of-the-art approaches in 2019 in general natural language understanding tasks [1]. Finally, we present our contributions which are composed of a new dataset for the monolingual phrase synonymy task and a new neural network architecture dedicated to encoding phrases (short sequences).

3.1 Approaches without learnable parameters

If we agree with "Everything should be made as simple as possible, but not simpler" (Albert Einstein), we would like to study the simplest approaches which often have already satisfying performance. We choose three simple approaches. The first one consists in the traditional

[1] https://gluebenchmark.com/leaderboard/

compositional stack approach which compares all the possible word combinations, while the second and the third one generate one single vector representing a multi-word. More concretely, the second approach is the addition based approach which simply sum up all the word vectors in a multi-word sequence. The third approach consists in concatenating all word vectors into one single large vector.

3.1.1 Compositional stack

The compositional stack approach is extremely naive and direct, we simply stack all the words or word vectors in a list and then compare them one by one. If words are stacked in this way then the comparison is done with the help of a supervised task-oriented list, for example, a word synonym list for the phrase synonymy task. Otherwise if it is the word vectors that we stack then we compare the vector similarity between each word vector in the list and all the candidate word vectors.

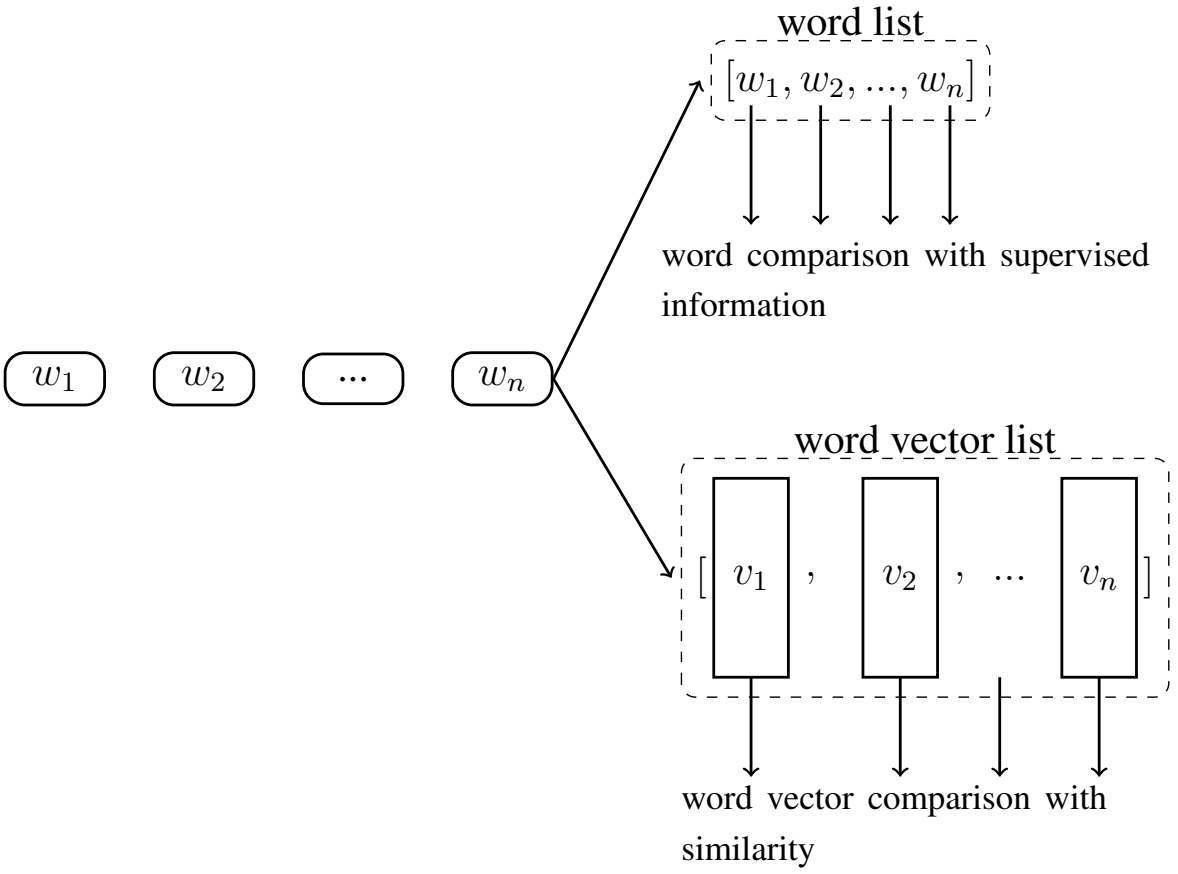


Figure 3.1 – Compositional stack.

For a given multi-word sequence $w_1, w_2, ..., w_n$ with n words, the word comparison looks at each of the words and replace it with a word of the given supervised list, e.g., a synonym. The word vector comparison replace each word with their corresponding word vector and map each token to the most similar word according to the vector similarity.

This simple approach may bring decent results if we have exhaustive task-oriented information such as a synonym word pair list. Nevertheless, it has several major drawbacks :

- Highly dependent on the task-oriented supervised information. If we do not possess such data, then the word comparison is inaccessible.
- Length sensitive. Even with the word vector comparison, this approach cannot deal with the fertility problem. We cannot obtain unified representation with this approach.
- Word inner relation ignored. This approach treats separately each word of a multi-word sequence, thus the inner relation between them is completely ignored. However we estimate that many multi-word expressions' meaning is deeply related to the association of words. This phenomenon is especially obvious when it comes to the idiomatic expressions or the homographs. For example, "a piece of cake" means in general *a job, task or other activity that is easy*, which is no more a part of cake. In "full magazine", the word "magazine" stands for *a store for arms or ammunition*, while in "fashion magazine" it means *a periodical publication*.

3.1.2 Addition based approach

One may argue that the most common baseline approach is the addition based approach. It is applicable when we use vector representations. For a given multi-word sequence $w_1, w_2, ..., w_n$ with n words, and their corresponding word vectors $v_1, v_2, ..., v_n$, the addition based approach simply use the sum as the vector representing the whole sequence :

$$v_{mw} = \sum_{i=1}^{n} v_i \tag{3.1}$$

where v_{mw} is the vector for representing the whole multi-word. *FastText* (Bojanowski et al., 2017) uses the same idea to obtain the word vectors with subword tokens. Some specify that this vector should be averaged by the sequence length n:

$$v_{mw} = \frac{\sum_{i=1}^{n} v_i}{n} \tag{3.2}$$

In practice, if we use the *cosine similarity*, for any two non-averaged vector x and y, they will have the same similarity as for their averaged version $\bar{x}$ and $\bar{y}$:

$$\begin{aligned} cos(\bar{x}, \bar{y}) &= \frac{\sum_{i=1}^{d} \frac{x_i}{a} \frac{y_i}{b}}{\sqrt{\sum_{i=1}^{d} (\frac{x_i}{a})^2} \sqrt{\sum_{i=1}^{d} (\frac{y_i}{b})^2}} \\ &= \frac{\sum_{i=1}^{d} \frac{x_i y_i}{ab}}{\frac{\sqrt{\sum_{i=1}^{d} {x_i}^2} \sqrt{\sum_{i=1}^{d} y_i^2}}{\sqrt{a^2}\sqrt{b^2}}} \\ &= \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2} \sqrt{\sum_{i=1}^{d} y_i^2}} = cos(x, y) \end{aligned} \tag{3.3}$$

where d is the word vector dimension size, a and b are the sequence length for the two multi-words.

The addition based approach has shown very strong results on sequence classification and similarity tasks (Mikolov, Sutskever, et al., 2013; Del et al., 2018; Liu et al., 2018). However, it has still two issues :

- Order ignored. For example, "service department" and "department service" will have the same representation while they do not have the same semantics.
- Word inner relation ignored. As described in Section 3.1.1.

3.1.3 Concatenation

The concatenation approach (Garten et al., 2015; Goikoetxea et al., 2016) consists in concatenating all the word vectors of a multi-word into one single high dimensional vector. If each word vector is of size d and the length of the multi-word is n, the concatenated vector will have a dimension size of nd.

The concatenation method does register word order but variable length phrases are no longer semantically comparable with the *cosine similarity* even if we pad them. For example the cosine similarity between "sneaker shoe" and "sneaker", denoted by $cos(v_{(sneaker,\ shoe)}, v_{(sneaker)})$ will

be $\frac{1}{2}$ if we pad "sneaker" with zeros. This similarity will be probably lower than :

$$cos(v_{(sneaker,\ shoe)}, v_{(sneaker,\ shop)})$$

However, it is obvious that "sneaker shoe" is semantically synonym of "sneaker". With this being said, we think the concatenation approach is not suitable for generating unified sequence representation even if it generates one single vector.

3.2 Phrase embeddings with CBOW or Skip-gram

Concerning the models with learnable parameters, many would wonder if it is possible that we simply train a word embedding model using CBOW or Skip-gram if we treat the n-grams as a single token. The answer is yes, we can indeed train a word embedding model just like what we do for all the n-grams. However, it ignores compositionality and inner word relations. Another way of training embedding model for multi-words is the extended Skip-gram introduced Artetxe, Labaka and Agirre (2018b). which is a generalized skip-gram that learns n-gram embeddings on-the-fly while keeping the desirable property of unigram invariance to handle compositional phrases but it still suffers from the sparsity and memory problem. In this section we will go through these two strategies for training a multi-word embedding model.

3.2.1 One single token processing for non-compositional phrases

Treating n-grams as a single token in a pre-processing step has been explored in the very inspiring work of Mikolov, Sutskever, et al. (2013). But they mainly focused on the phrases whose meaning is not a simple composition of the meanings of its individual words, such as *New York Times* and *Toronto Maple Leafs*. They proposed to extract some n-grams based on a threshold score which is defined as :

$$score(w_i, w_j) = \frac{n(w_i, w_j)\delta}{n(w_i)n(w_j)} \tag{3.4}$$

where $n(w)$ is the count for the word w and δ is used as a discounting coefficient and prevents too many phrases consisting of very infrequent words to be formed. The bigrams with score above the chosen threshold score are extracted and after 2-4 passes over the training data with decreasing threshold value, they build recursively n-grams longer than 2 tokens.

Clearly, this approach works relatively well on more idiomatic phrases, whereas in our scenario, most of the phrases are freely combined multi-words. In addition, this approach can

only handle phrases that have been passed into the model during the training. A new phrase, even if it is idiomatic, can never have its phrase vector through this approach.

3.2.2 Extended skip-gram with negative sampling

Apart from the more idiomatic phrases extracted by the score of Equation 3.4, to learn the embeddings for other n-grams which are generally more compositional phrases, one could simply merge all n-grams but this would greatly increase the vocabulary size, making the model time and space too complex. This idea has shown very poor performance in the preliminary experiments of Artetxe, Labaka and Agirre (2018b) while they tried to randomly generate multiple consistent segmentations for each sentence.

In order to produce relevant phrase embeddings with Skip-gram, Artetxe, Labaka and Agirre (2018b) propose to extend the *Skip-gram* with *negative sampling* model. Recall the loss function of this model in Equation 2.50, for the ease of the reading, we copy it here :

$$E = -\log \sigma(s_{w_O}^T h) - \sum_{w_k \in \mathcal{W}_{neg}} \log \sigma(-s_{w_k}^T h) \tag{3.5}$$

where h is the output of the hidden layer for the input word token w, s is the output of the hidden layer for the context $c \in w_O \cup \mathcal{W}_{neg}$. This can be seen as a logistic regression to predict whether the word-context pair really co-occurs in the on-line sampled sentence, or the context is one of the negative samples in $\mathcal{W}_{neg}$. Updating the hidden weight matrix W_2 is equal to updating the word vector w and context vectors c.

In the extended *Skip-gram* with *negative sampling*, in addition to the updates based on the single-word pairs (update the vector of w and all the vectors of c), they also extract all the possible n-grams phrases, p, within a max length and update all the vectors of p while using the same context unigrams c.

The positive and negative context vectors in W_2 are only updated for single-word pairs (w, c). The **update function** is the same as in Equation 2.52 because the positive and negative contexts are always unigrams. For each n-gram p, the corresponding embedding vector is directly updated in W_1 using the n-gram and context single-word pairs (except that the vectors in W_2 are not updated this time). According to the authors, this allows to naturally learn n-gram embeddings based on their co-occurrence patterns as modeled by Skip-gram, without introducing subtle interactions that affect its fundamental behavior.

In conclusion, this approach learns all the n-grams in a corpus including the compositional phrases. But it still cannot generate a phrase embedding for phrases that do not appear in the

training corpus.

3.3 Traditional neural networks for sequential inputs

We have seen the "simple approaches" for modeling multi-word representations. All these approaches ignore the inner structure of multi-words. In this section we discuss some more complex approaches using neural networks for sequential inputs. We will first go through three types of these neural networks : the *Convolutional Neural Network* (CNN), the *Recurrent Neural Network* (RctNN) and its variants such as *Long short-term memory* (LSTM) and *Gated Recurrent Unit* (GRU). Finally, we present the latest application: *Embeddings from Language Models* (ELMo) using the bidirectional *LSTM* as basic unit.

3.3.1 Convolutional Neural Network

The *Convolutional Neural Network* (CNN) is also a *feed forward neural network*. Inspired by the *Receptive Field* theory in neural science (Hubel and Wiesel, 1968), Fukushima (1980) introduced the basic layer using the convolution operation.

Convolution is a mathematical operation, denoted by $\circledast$, here we only consider the discrete case. Let f and g be two functions defined on the integers, the 1-d discrete convolution of f and g is :

$$(f \circledast g)[n] = \sum_{m=-inf}^{inf} f[m]g[n-m] \tag{3.6}$$

and the 2-d discrete convolution:

$$(f \circledast g)[i,j] = \sum_{k=-inf}^{inf} \sum_{t=-inf}^{inf} f[k,t]g[k-i,t-j] \tag{3.7}$$

In practice, we let our input distribution as one function defined on indices (discrete values), and a *kernel* (or *filter*, containing some weights) as the other function. The idea is to repeatedly apply the kernel on all the outputs. The convolution output is also called **feature map** because the convolution can be seen as a feature extraction process. Usually, the definition domain of the kernel is much narrower. For the 1-d convolution, we can note the input x_i with $i \in [1,n]$, the kernel function f_i with $i \in [1,m]$ where $m << n$. The output sequence y_i is :

$$y_i = \sum_{k=1}^{m} f_k x_{i-k+1} \tag{3.8}$$

Figure 3.2 shows an example of the 1-d convolution.

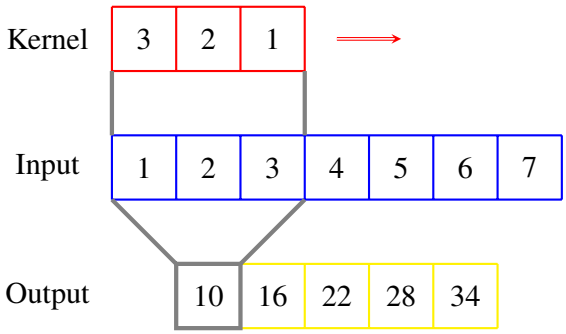


Figure 3.2 – 1-d convolution with kernel length $m = 3$ for an input sequence with length $n = 7$. The output sequence length is $n - m + 1$.

As for the 2-d convolution which is widely applied in the *computer vision* domain, we note the input graph $x_{i,j}$ with $i \in [1, n]$, $j \in [1, q]$, the kernel $f_{i,j}$ with $i \in [1, m]$, $j \in [1, p]$ where $m << n$ and $p << q$. The output graph $y_{i.j}$ is then :

$$y_{i,j} = \sum_{u=1}^{m} \sum_{v=1}^{p} f_{u,v} x_{i-u+1,j-v+1} \tag{3.9}$$

Figure 3.3 shows an example of the 2-d convolution.

In fact, we can move the kernel by a larger **stride**, in the illustrated examples the stride is always 1, the output size after the convolution is $n - f + 1$ where n is the input size and f the kernel size at the corresponding dimension. Apart from stride, we can also **zero-pad** the input. With the stride and zero-padding into account, the output size is $\frac{n-f+2padding}{stride} + 1$. Note that if the result is not an integer, then the current convolution does not fit, we should always make sure that the convolution process can be successfully completed.

Compared to the fully connected neural network, a convolutional neural network can be considered as a partially connected neural network. In the fully connected network, n^l is the dimension size at layer l, the weight matrix between layer $l - 1$ and l will have $n^l \times n^{l-1}$ parameters. Each neuron in layer $l - 1$ is connected to n^l weights, whereas in the convolutional network, each neuron in layer $l - 1$ is only connected to a window of weights. The passage function between layer $l - 1$ and l for 1-d convolution is :

$$a^l = f(w^l \circledast a^{l-1} + b^l) \tag{3.10}$$

where $w^l \in \mathbb{R}^m$ is the kernel. We only need m weight parameters for this layer. However,

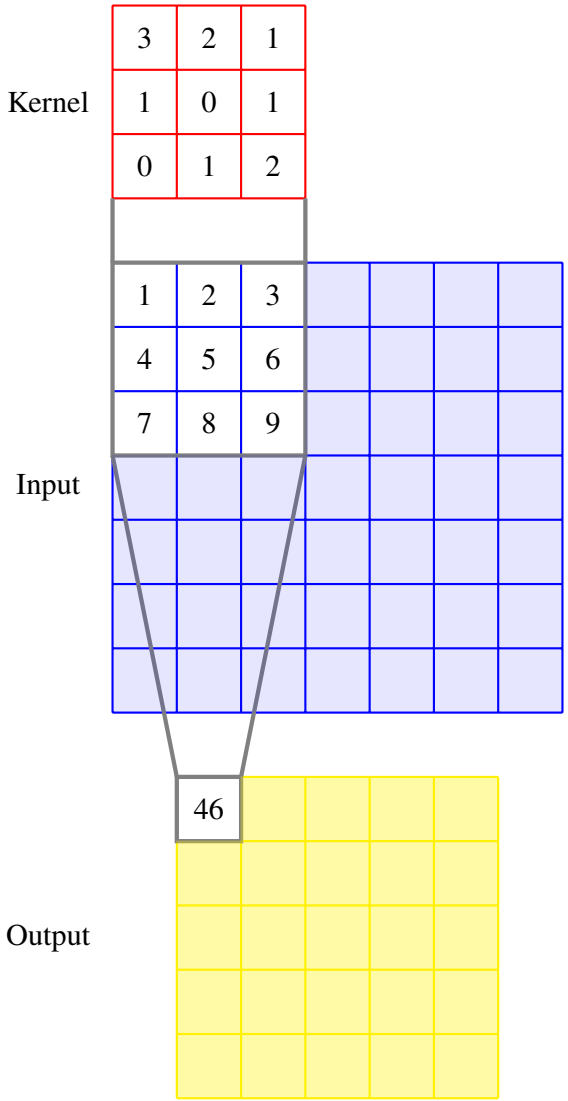


Figure 3.3 – 2-d convolution with a 3×3 kernel graph and for a 7×7 input graph. The output sequence length is $(7 - 3 + 1) \times (7 - 3 + 1)$. We only show one output value, the rest can be calculated by sliding the kernel on the input graph.

unlike the fully connected layer, the output neuron number of each layer is not configurable. It is determined by the kernel size m :

$$n^l = \frac{n^{l-1} - m + 2padding}{stride} + 1 \tag{3.11}$$

Similarly, for the 2-d convolution, the graph X^{l-1} in layer $l-1$ is transformed to X^l in layer l following :

$$X^l = f(W^l \circledast X^{l-1} + B^l) \tag{3.12}$$

where $W^l \in \mathbb{R}^{u*v}$ is the kernel, $X^l \in \mathbb{R}^{w_l * h_l}$ and $X^{l-1} \in \mathbb{R}^{w_{l-1} * h_{l-1}}$ are the graph in layer l and $l-1$. The size of graph X^l is :

$$\begin{aligned} w^l &= \frac{w^{l-1} - u + 2padding}{stride} + 1 \\ h^l &= \frac{h^{l-1} - u + 2padding}{stride} + 1 \end{aligned} \tag{3.13}$$

In order to learn multiple aspects from the input, we can apply several different kernels, the number of kernels is called **depth**. Usually, we add a **pooling** or *subsampling* layer after obtaining the outputs of each kernel to reduce the final output dimension and thus avoid the overfitting problem. Pooling layers downsample each feature map independently, reducing the height and width, keeping the depth intact.

Contrary to the convolution operation, pooling has no weight parameters. We only have to set the stride and window hyper-parameters. The most common type of pooling is the *max pooling* and the *average pooling*:

$$\begin{aligned} pool_{max}(R_k) &= \sum_{i \in R_k} \max(a_i) \\ pool_{avg}(R_k) &= \frac{1}{|R_k|} \sum_{i \in R_k} a_i \end{aligned} \tag{3.14}$$

where R_k is the pooling window and a_i the feature map value at position i. Figure 3.4 shows an example of a max pooling with a stride of 1 and a 2×2 pooling layer.

We also show an illustration of the famous *LeNet-5* for handwritten number recognition which is built on some convolutional layers (Lecun et al., 1998) with a depth column in each layer (with multiple kernels) :

The **gradients** calculation is fairly straightforward with the formula given in 2.26. For the $\delta^{(l,i)}$ which is the error in the layer l for the i-th value of the feature map, if the pooling layer is

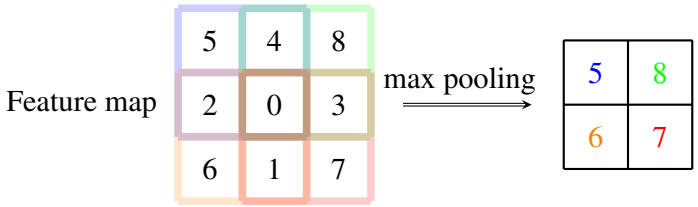


Figure 3.4 – An example of max pooling.

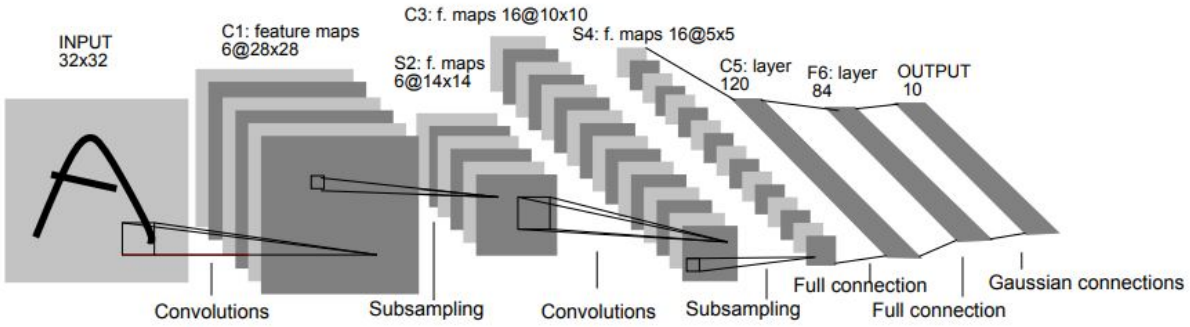


Figure 3.5 – LeNet-5 architecture. Original illustration from Lecun et al. (1998).

the $l+1$-th layer, then we have :

$$\begin{aligned}\delta^{(l,i)} &= \frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial Z^{(l,i)}} \\ &= \frac{\partial X^{(l,i)}}{\partial Z^{(l,i)}} \boxed{\frac{\partial Z^{(l+1,i)}}{\partial X^{(l,i)}}} \frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial Z^{(l+1,i)}} \\ &= f_l'(Z^{(l,i)}) \odot \boxed{\mathbf{up}}(\delta^{(l+1,i)})\end{aligned} \tag{3.15}$$

where Y and $\hat{Y}$ are respectively the output and real label of the network. X^l is the final output of the layer l after activation or the input of the layer $l+1$, Z^l is the output of the layer l before the activation. Since the $l+1$-th layer is a pooling layer, Z^{l+1} is the output of the pooling with the input X^l. The **up** (*upsampling*) is the reverse function of the *subsampling*, for instance, for the *max pooling*, $\mathbf{up}(\delta^{(l+1,i)})$ will simply associate every value to the maximum value of the feature map in the previous layer. The rest positions will be 0. While for the *average pooling*, the values in $\delta^{(l+1,i)}$ will be equally distributed to every position of the feature map in the previous layer.

Similarly, if the pooling layer is the l-th layer and the $l+1$-th layer is a convolutional layer,

the i-th feature map for layer $l+1$, $Z^{(l+1,i)}$ is :

$$Z^{(l+1,i)} = \sum_{d=1}^{D} W^{(l+1,i,d)} \circledast X^{(l,d)} + b^{(l+1,i)} \tag{3.16}$$

where D is the convolution depth. Then the d-th error in the l-th layer $\delta^{(l,d)}$ is :

$$\begin{aligned}
\delta^{(l,d)} &= \frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial Z^{(l,d)}} \\
&= \frac{\partial X^{(l,d)}}{\partial Z^{(l,d)}} \frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial X^{(l,d)}} \\
&= f_l'(Z^l) \odot \sum_i (\textbf{rot180}(W^{(l+1,i,d)}) \circledast \frac{\partial \mathcal{L}(Y,\hat{Y})}{\partial Z^{(l+1,i)}}) \\
&= f_l'(Z^l) \odot \sum_i (\textbf{rot180}(W^{(l+1,i,d)}) \circledast \delta^{(l+1,i)})
\end{aligned} \tag{3.17}$$

where **rot180** is a rotation of 180 degrees. The detail of the derivative calculation for the convolution operation is omitted here.

In our work, we use 1-d convolution with zero-padding to encode our phrases as one of the baseline approaches.

3.3.2 Recurrent Neural Network

In the feedforward network, the information is transferred in a uni-directional way. The layer output depends only on the input, this grants the network a quick learning of the parameters but in many actual tasks, one neuron's output also depends on the "memory" or previous outputs. *Recurrent neural network* (RctNN) is in line with this need, it is capable of memorizing a short period by taking not only the current input but also the previous output of itself.

Figure 3.6 shows the basic idea of RctNN, for a sequence $x_t, t \in [1, T]$, the RctNN transfers the information by the following formula :

$$\begin{aligned}
h_t &= f(h_{t-1}, x_t) \\
\hat{y}_t &= g(h_t)
\end{aligned} \tag{3.18}$$

where h_t is the current hidden state and $\hat{y}_t$ is the output at the current time step t. f and g are two non-linear functions, i.e., a feedforward network.

The *simple recurrent neural network* (SRNN) (Elman, 1990) is a basic recurrent neural

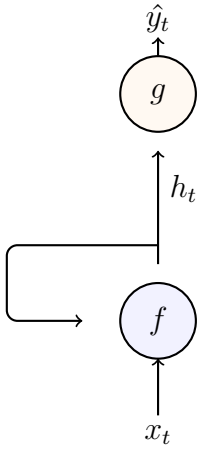


Figure 3.6 – A basic cell of RctNN.

network architecture with only one hidden layer. Each net input z_t[2] at time step t of the hidden layer is calculated based on the current network input x_t :

$$\begin{aligned} z_t &= Uh_{t-1} + Wx_t + b \\ h_t &= f(z_t) \end{aligned} \tag{3.19}$$

where U and W are weight matrices and b the bias vector. If we consider that each time step is a single layer, then RctNN is actually a multiple layer network with shared weights. We can unfold in Figure 3.6 :

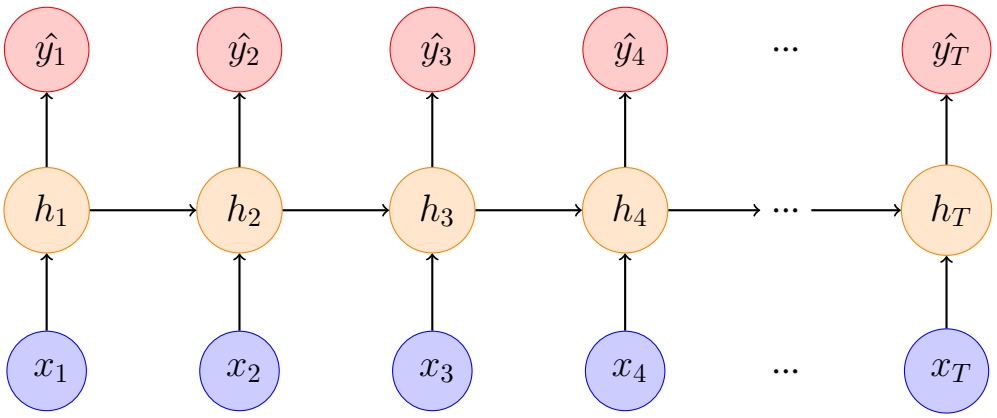


Figure 3.7 – Unfolded RctNN

Now we look at the parameter learning with **gradient** descent, the loss function at time step

[2]We refer *net input* or *net activation* to the layer output before the activation function

t is :

$$\mathcal{L}_t = \mathcal{L}(y_t, g(h_t)) \tag{3.20}$$

The loss function for all time steps is :

$$\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}_t \tag{3.21}$$

The gradient for U :

$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial U} \tag{3.22}$$

Since U is the weight matrix related to the previous hidden state, the gradient calculation is different from the feedforward networks. Typically, we can use the *Back propagation through time* (BPTT) (Werbos, 1990) to obtain gradients in a RctNN.

The BPTT treats the RctNN as an unfolded multi-layer feedforward network as shown in Figure 3.7. In this unfolded multi-layer network, all the layers share the same parameters, so the gradients are the sum of the gradients in each layer. We begin with the derivative in Equation 3.22. Because the derivative at time step t is related to all the previous net inputs z_k, with $k \in [1, t]$, we have:

$$\frac{\partial \mathcal{L}_t}{\partial u_{i,j}} = \sum_{k=1}^{t} \frac{\partial z_k(U)}{\partial u_{i,j}} \frac{\partial \mathcal{L}_t}{z_k} \tag{3.23}$$

Since $z_k = Uh_{k-1} + Wx_k + b$, the direct derivative of z_k with regard to $u_{i,j}$, denoted by $z_k(U)$ which means that we consider z_k as a function on U with h_{k-1} as a constant :

$$\frac{\partial z_k(U)}{\partial u_{i,j}} = \mathbb{I}_i(h_{k-1}[j]) \tag{3.24}$$

where $\mathbb{I}_i(h_{k-1}[j])$ means a vector with the same size as h_{k-1} with all values set to zero except for the i-th dimension whose value is equal to $h_{k-1}[j]$.

For the second part in 3.23, again we can define an error $\delta_{t,k} = \frac{\partial \mathcal{L}_t}{\partial z_k}$ which is the derivative

of the network loss at time step t with regard to the net input at time step k.

$$\begin{aligned}\delta_{t,k} &= \frac{\partial \mathcal{L}_t}{\partial z_k} \\ &= \frac{\partial h_k}{\partial z_k}\frac{\partial z_{k+1}}{\partial h_k}\frac{\partial \mathcal{L}_t}{\partial z_{k+1}} \\ &= \mathbf{diag}(f'(z_k))U^T\delta_{t,k+1}\end{aligned} \tag{3.25}$$

So the formula in 3.23 can be written as:

$$\begin{aligned}\frac{\partial \mathcal{L}_t}{\partial u_{i,j}} &= \sum_{k=1}^{t}\delta_{t,k}[i]h_{k-1}[j] \\ \frac{\partial \mathcal{L}_t}{\partial U} &= \sum_{k=1}^{t}\delta_{t,k}h_{k-1}^T\end{aligned} \tag{3.26}$$

So with substitution in 3.22, the **gradient** for the total loss with regard to U is:

$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{T}\sum_{k=1}^{t}\delta_{t,k}h_{k-1}^T \tag{3.27}$$

For the same reason, the **gradients** for other parameters are:

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T}\sum_{k=1}^{t}\delta_{t,k}x_k^T \tag{3.28}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{t=1}^{T}\sum_{k=1}^{t}\delta_{t,k} \tag{3.29}$$

Figure 3.8 gives an illustration of BPTT.

Another way of calculating the gradients is *real-time recurrent learning* or RTRL (Williams and Zipser, 1995) which is a forward procedure similar to the forward mode in auto-differentiation (See 2.2.1). RTRL follows the same gradient calculation except we calculate each δ in the same order as the time sequence and update each parameter right after obtaining the corresponding gradient.

The hidden state at time step $t+1$, h_{t+1} is:

$$h_{t+1} = f(z_{t+1}) = f(Uh_t + Wx_{t+1} + b) \tag{3.30}$$

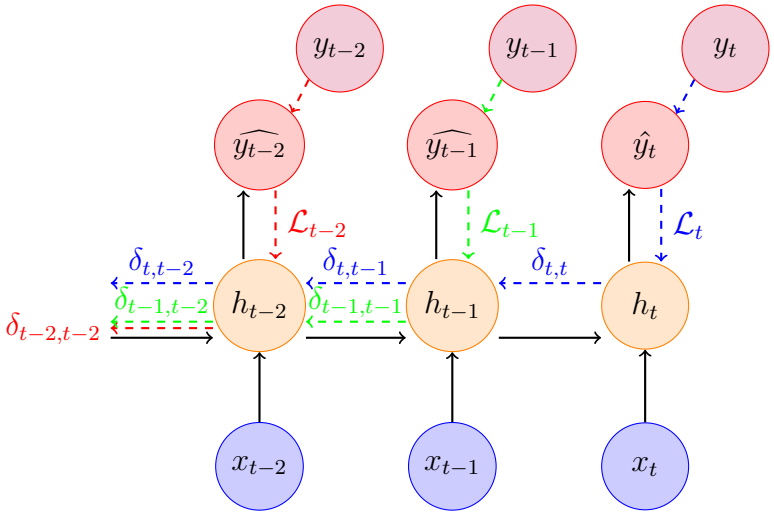


Figure 3.8 – Diagram of BPTT. Black lines are the forward pass and dashed lines are the local backward pass. Gradients are accumulated through time.

The derivative of h_{t+1} with regard to $u_{i,j}$ is:

$$
\begin{aligned}
\frac{\partial h_{t+1}}{\partial u_{i,j}} &= \boxed{\frac{\partial z_{t+1}}{\partial u_{i,j}}} \frac{\partial f(z_{t+1})}{\partial z_{t+1}} \\
\boxed{\frac{\partial z_{t+1}}{\partial u_{i,j}}} &= \frac{\partial U h_t}{\partial u_{i,j}}, \text{ and because } \frac{\partial f(x)g(x)}{\partial x} = f'(x)g(x) + g'(x)f(x), \text{ above line equals to} \\
&= \boxed{(\frac{\partial U}{\partial u_{i,j}} h_t + \frac{\partial h_t}{\partial u_{i,j}} U^T)} \frac{\partial f(z_{t+1})}{\partial z_{t+1}} \\
&= (\mathbb{I}_i(h_t[j]) + \frac{\partial h_t}{\partial u_{i,j}} U^T) \mathbf{diag}(f'(z_{t+1})) \\
&= (\mathbb{I}_i(h_t[j]) + \frac{\partial h_t}{\partial u_{i,j}} U^T) \odot (f'(z_{t+1}))^T
\end{aligned}
\tag{3.31}
$$

We use Equation 3.31 to calculate $\frac{\partial h_1}{\partial u_{i,j}}$, $\frac{\partial h_2}{\partial u_{i,j}}$, $\frac{\partial h_3}{\partial u_{i,j}}$, ... in a forward order and at each time step t, the gradient can be obtained simultaneously with the current loss $\mathcal{L}_t$:

$$
\frac{\partial \mathcal{L}_t}{\partial u_{i,j}} = \frac{\partial h_t}{\partial u_{i,j}} \frac{\partial \mathcal{L}_t}{\partial h_t} \tag{3.32}
$$

Note that in RTRL we will update the parameters right after obtaining the gradient at each step. The update for W and b follows the same way.

In general, as we usually use sigmoid functions as the activation function and its derivative is

inferior to 1, the sequence product of multiple derivatives of sigmoids functions would approach 0, causing the **gradient vanishing** problem for $\frac{\mathcal{L}_t}{\partial h_k}$ where $t - k$ is relatively high[3]. In other words, the hidden states h_k which are far from the current step t would merely influence the current gradient $\frac{\mathcal{L}_t}{\partial U}$, the RctNN can only memorize short distance dependencies.

Concerning our work, we also implement RctNN as a baseline approach for phrase representation modeling.

3.3.3 LSTM and GRU

To overcome the **gradient vanishing** problem for long distance dependencies, we could add linear relation between each time step so there will be both linear and non-linear relations and it would be less possible for the gradient to vanish. However, there are still problems such as **gradient explosion** and limited **memory capacity**[4]. In order to solve these problems, some RctNN variants are proposed with a **gate** mechanism. We will cover the two most successful gated RctNN: *Long Short-Term Memory* (LSTM) (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) and *Gated Recurrent Unit* (GRU) (Cho et al., 2014; Chung et al., 2014).

Long Short-Term Memory

In addition to the *hidden state* or *external state* h_t in the RctNN, LSTM introduces an *internal state* or *cell state* c_t (with a candidate cell $\tilde{c}_t$) and three **gates** whose range is $[0, 1]$ in order to indicate how much information should pass the "gate":

- **Forget gate** f_t decides how much information in the previous cell c_{t-1} should drop:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3.33}$$

- **Input gate** i_t decides how much information in the current candidate cell $\tilde{c}_t$ should be passed:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.34}$$

- **Output gate** o_t decides how much information in the current cell c_t should be passed to the current hidden h_t:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3.35}$$

[3] Similarly, if the derivative is superior to 1, it will cause the **gradient explosion** problem.

[4] h_t would converge with information accumulated through time. The memory of h_t is limited.

We can see that the three gates are all functions about the current input x_t and the previous hidden state vector h_{t-1}, with a logistic function to obtain values between $[0, 1]$ in order to control the information passing proportion.

Basically, one LSTM layer passes information with the hidden and cell state:

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{3.36}$$

where the candidate cell $\tilde{c}_t$ is:

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3.37}$$

The LSTM architecture is shown in Figure 3.9 in a computational graph form, we can consider an LSTM unit as a special RctNN unit, with two input and output states c and h.

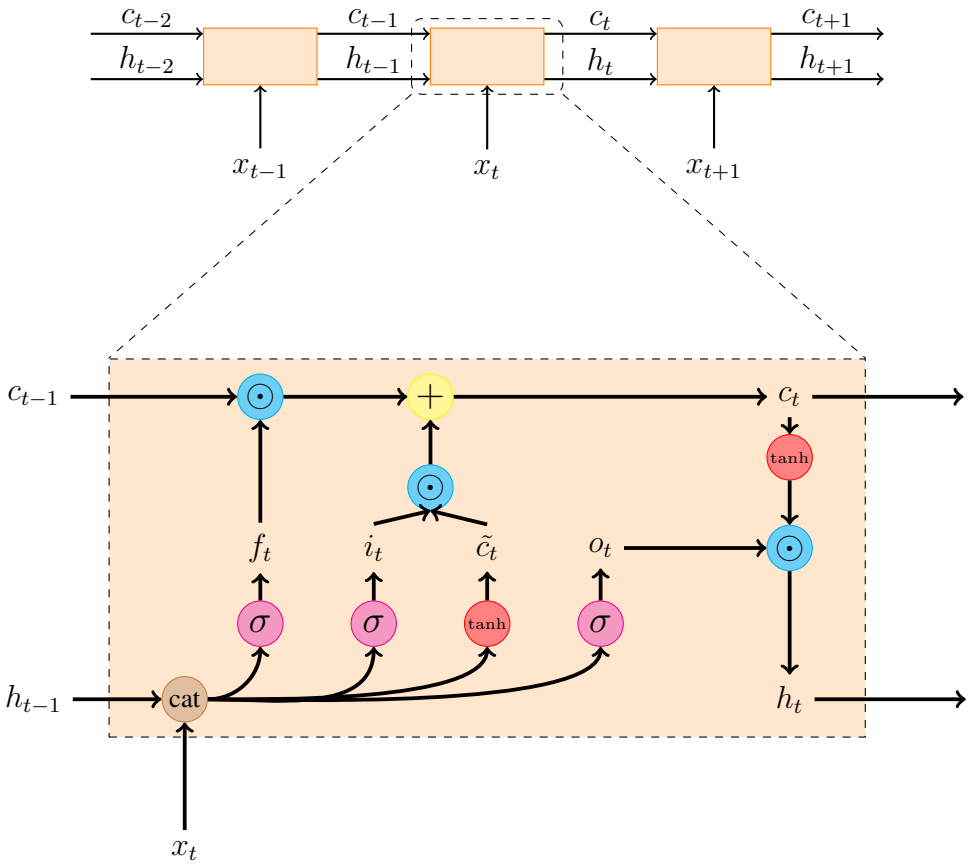


Figure 3.9 – Diagram of LSTM.

In practice, we can concatenate the input $x_t \in \mathbb{R}^a$ and the previous hidden $h_{t-1} \in \mathbb{R}^d$, so we can write one single weight matrix for the matrices in Equation 3.33, 3.34, 3.35 and 3.37: $W \in \mathbb{R}^{d*(a+d)}$. We usually set relatively high initial values for the parameters in the forget gate since otherwise most of the previous information would be dropped[5].

The network can also be learned by gradient descent and the gradient calculation is not very different from what we have seen in RctNN. We will not go through all the details for LSTM and all the advanced network architecture in later sections.

There are also many variants of LSTM, the work of Greff et al. (2017) and Jozefowicz et al. (2015) has compared a good deal of popular variants and concluded that they are essentially the same architecture and have similar experimental performance.

In a neural network, **long-term memory** consists in the network parameters, which covers all the training period and are updated gradually. In a LSTM network, the cell state c is capable of storing and transferring information for a short period yet still longer than the hidden state h, that is why we call the network **Long short-term memory**.

We will also include LSTM in our work as a baseline approach for encoding phrases.

Gated Recurrent Unit

Gated recurrent unit (GRU) is simpler than LSTM, it does not introduce a new state like the cell state in LSTM. Still, GRU uses several "gates": an **update gate** which is a combination of the forget and input gates in LSTM and a **reset gate** which determines the dependency on the previous hidden state h_{t-1}.

- **Update gate** z_t. It decides how much information should be kept from the previous hidden h_{t-1} and meanwhile, how much information should be passed from the current candidate state $\tilde{h_t}$.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{3.38}$$

- **Reset gate** r_t. It decides the degree of the dependency on the previous hidden h_{t-1} for the current candidate state $\tilde{h_t}$.

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{3.39}$$

The current candidate hidden state $\tilde{h_t}$ is defined as :

$$\tilde{h_t} = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{3.40}$$

[5]Generally speaking, the initial values for the parameters in a neural network are quite small, here in LSTM, we would like to set a high value for the bias vector in the forget gate, for instance b_f = 1

Then the information in the GRU network is passed with the hidden state h_t :

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \tag{3.41}$$

From 3.41 we can see that when $z_t = 1$, the current hidden state h_t is linearly related to the previous hidden state h_{t-1}, and when $z_t = 0$, h_t is related to h_{t-1} in a non-linear manner as $\tilde{h}_t$ is obtained from a sigmoid function $\tanh$. From 3.40 we can see that when $r_t = 1$, the current candidate hidden state $\tilde{h}_t$ depends on both the current input x_t and the previous hidden state h_{t-1}, and when $r_t = 0$, $\tilde{h}_t$ depends on only x_t. Combining these observations, when $z_t = 0, r_t = 1$, the GRU network is basically a recurrent neural network. Furthermore, when $z_t = 0, r_t = 0$, the current hidden h_t is only related to the current input x_t, which is similar to one separated fully connected layer. Figure 3.10 illustrates how GRU passes information.

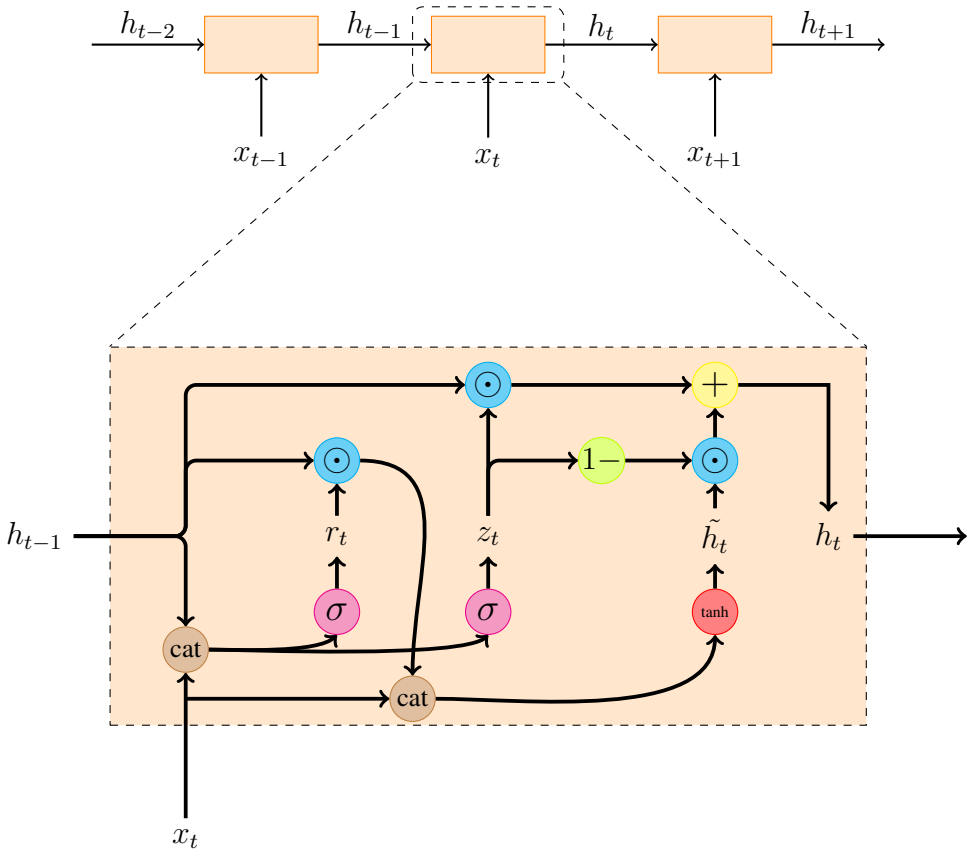


Figure 3.10 – Diagram of GRU.

Since GRU has very similar performance with LSTM, we will not include GRU in our frameworks, one can refer to LSTM to get an approximate performance of GRU.

3.3.4 Recent applications: ELMo

There are many great works incorporating a recurrent neural network or its variants. Quite recently, *embedding from language models* (ELMo) (Peters et al., 2018) has been proposed which is built on a deep bidirectional LSTM network (BiLSTM) with subword information encoded by a character convolutional network. The network is first trained on non-supervised bidirectional language modeling, then the pre-trained model can be applied to some supervised NLP tasks by adding some other layers on top while freezing the weights in the pre-trained model[6].

Bidirectional language model with LSTMs

Similarly to the translation objective in Equation 1.1, given a sequence of T tokens, $x^k, k \in [1, T]$, a forward language model computes the probability of the sequence by modeling the probability of the current token x^k given all the previous tokens $(x^1, x^2, ...x^{k-1})$:

$$P(x^1, x^2, ..., x^T) = \prod_{k=1}^{T} P(x^k | x^1, x^2, ..., x^{k-1}) \tag{3.42}$$

Analogously, a backward language model is a reversed version of the forward one :

$$P(x^1, x^2, ..., x^T) = \prod_{k=1}^{T} P(x^k | x^T, x^{T-1}, ..., x^{k+1}) \tag{3.43}$$

Each token is represented by a vector v_k (input embedding), and it is passed to L stacked layers of forward and backward LSTMs, obtaining $\overrightarrow{h_{k,j}}$ and $\overleftarrow{h_{k,j}}$ where $j \in [1, L]$ means the j-th LSTM layer. The top LSTM layer's outputs $\overrightarrow{h_{k,L}}$ and $\overleftarrow{h_{k,L}}$ are used to predict the next token x_{t+1} with a softmax function.

$$\begin{aligned} \overrightarrow{h_{k,j}} &= \overrightarrow{\text{LSTM}_j}(v_k, \overrightarrow{h_{k-1,j}}) \\ \overleftarrow{h_{k,j}} &= \overleftarrow{\text{LSTM}_j}(v_k, \overleftarrow{h_{k+1,j}}) \end{aligned} \tag{3.44}$$

A bidirectional language model (biLM) maximizes the log likelihood:

$$\sum_{k=1}^{T} (\log P(x^k | x^1, x^2, ..., x^{k-1}; \overrightarrow{\theta}) + \log P(x^k | x^T, x^{T-1}, ..., x^{k+1}; \overleftarrow{\theta})) \tag{3.45}$$

where θ means the network parameters.

Linear combination of biLM

For each token x_k, we can conclude that a L-layer biLM contains a set of $2L + 1$ vector

[6] We can conclude this manner as a *feature extraction*, further discussion can be found in the next chapter.

Task	Input Only	Input & Output	Output Only
SQuAD	85.1	**85.6**	84.8
SNLI	88.9	**89.5**	88.7
SRL	**84.7**	84.3	80.9

Figure 3.11 – Development set performance for SQuAD, SNLI and SRL when including ELMo at different locations in the supervised model. Extracted from the original paper of Peters et al. (2018). The data set details can be found in the original paper.

representations:

$$\begin{aligned} R_k &= \{v_k, \overrightarrow{h_{k,j}}, \overleftarrow{h_{k,j}} | j \in [1, L]\} \\ &= \{v_k, \overleftrightarrow{h_{k,j}} | j \in [1, L]\} \end{aligned} \tag{3.46}$$

where $\overleftrightarrow{h_{k,j}}$ means the concatenated vector of $[\overrightarrow{h_{k,j}}; \overleftarrow{h_{k,j}}]$.

In ELMo, the outputs in each layer are collapsed into a single vector $ELMo_k = E(R_k; \theta)$. For a specific task, ELMo computes different weights for each layer:

$$ELMo_k^{task} = E(R_k; \theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \overleftrightarrow{h_{k,j}} \tag{3.47}$$

where s_j^{task} are softmax-normalized weights in terms of layer and task, and γ^{task} is a scalar parameter which allows the model to scale the ELMo vector. Equation 3.47 can also be seen as a linear combination of each BiLM layer.

Integration of ELMo

ELMo vectors can be incorporated into other supervised NLP models. Basically, the author state that ELMo vector can substitute static word embedding vectors such as CBOW, Skip-gram or GloVe. For instance, to integrate ELMo into a task-specific RctNN model, we can first freeze the weights of the BiLM and then pass the concatenated vector $[v_k; ELMo_k^{task}]$ obtained from ELMo vector $ELMo_k^{task}$ and the input vector v_k into the RctNN. For some tasks, authors find that it is beneficial to integrate ELMo vectors at the output layer of a RctNN which means replacing the hidden state h_k with $[h_k; ELMo_k^{task}]$. Or sometimes, we can make these modifications at the same time, changing the input and the output of a RctNN. In the ablation tests, authors have shown some interesting results about different application locations of ELMo vectors.

Besides, in addition to using all the layers, we can use one particular layer of the biLM.

Task	Baseline	Last Only	All layers	
			λ=1	λ=0.001
SQuAD	80.8	84.7	85.0	**85.2**
SNLI	88.1	89.1	89.3	**89.5**
SRL	81.6	84.1	84.6	**84.8**

Figure 3.12 – Development set performance for SQuAD, SNLI and SRL comparing using all layers of the biLM (with different choices of regularization strength λ) to just the top layer. Extracted from the original paper of Peters et al. (2018). The data set details can be found in the original paper.

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 $\pm$ 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 $\pm$ 0.19	90.15	92.22 $\pm$ 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 $\pm$ 0.5	3.3 / 6.8%

Figure 3.13 – Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F1 for SQuAD, SRL and NER; average F1 for Coref. The "increase" column lists both the absolute and relative improvements over the baseline. Extracted from the original paper of Peters et al. (2018). The data set details can be found in the original paper.

In Figure 3.12, ablation tests conducted by authors show that using all the layers does help the model to encode different types of syntactic and semantic information.

Conclusion

We report the main results of ELMo vector in Figure 3.13.

In general, ELMo has strong generalization ability of modeling language, this has been proven by large improvements when applying ELMo to a broad range of NLP tasks. Therefore, we would like to not only include ELMo for a comparison with our systems but also make our systems capable of integrating ELMo as an input layer.

3.4 Transformers: Multi-Head Attention

The neural networks for processing sequential inputs must have a large memory capacity in order to stock more information, which requires enlarging the network's complexity. In this section, we begin with presenting the attention mechanism which manages to focus on certain aspects of the input information, reducing the network's input size. Then we study a recent attention architecture called ***multi-head attention*** (Vaswani et al., 2017) which has constructed the **Transformer** architecture, achieving strong empirical results, many new state-of-the-art architectures are derived from the Transformer.

3.4.1 Attention mechanism

In the past few years, attention mechanisms have brought effective progression to many NLP tasks. Basically they can be considered as a data/task-driven weighting process for the input information. For each input in a sequence, the attention applies a weight indicating how much "attention" we should pay to the input. Bahdanau et al. (2014) use an attention-based model for neural machine translation with an encoder-decoder architecture, Xu et al. (2015) generalize attention into *soft* and *hard* attention while Luong, Pham, et al. (2015b) further develop them into *global attention* and *local attention*. Ling et al. (2015) employ attention to obtain better word representations with respect to word contexts.

Generally speaking, attention mechanisms can be split into two steps, the first one consists in calculating the attention distribution and the second step calculates the weighted mean.

Attention distribution

Given a sequence of input vector $v_i, i \in [1, N]$, the attention distribution is obtained by applying the softmax function on a score on the input and a **query vector** q.

$$\begin{aligned} \alpha_i &= \text{softmax}(s(v_i, q)) \\ &= \frac{\exp(s(v_i, q))}{\sum_j^N \exp(s(v_i, q))} \end{aligned} \tag{3.48}$$

where α_i is the attention distribution for the i-th input vector and s is a score function which can be obtained by one of the following formulas:

- Addition.

$$s(v_i, q) = s^T \tanh(Wv_i + Uq) \tag{3.49}$$

- Dot-product.

$$s(v_i, q) = v_i^T q \tag{3.50}$$

- Scaled dot-product.

$$s(v_i, q) = \frac{v_i^T q}{\sqrt{d}} \tag{3.51}$$

- General.

$$s(v_i, q) = v_i^T W q \tag{3.52}$$

where W, U, s are learnable parameters, d is the dimension size of the input vector v_i.

Weighted mean

The final attention is actually a weighted mean with the attention distribution α_i as the weight for the i-th vector:

$$\text{attention}(V, q) = \sum_{i=1}^{N} \alpha_i v_i \tag{3.53}$$

Equation 3.53 is also called the *soft attention* or *global attention*. The soft attention calculates its value depending on all the input vectors. In addition to the soft attention, *hard attention* also calculates the attention depending only on one input at one step (Xu et al., 2015).

$$\text{attention}(V, q) = v_j \tag{3.54}$$

where we copy one input v_j with j the index of the element with the highest probability, $\underset{i=1}{\overset{N}{\text{argmax}}}\, \alpha_i$ in the first formular of 3.48. The **soft attention** is smooth and differentiable but can be very expensive when the input is large. The **hard attention** is less complex at the inference time but is non-differentiable and requires more complicated techniques such as variance reduction or reinforcement learning to train (Luong, Pham, et al., 2015b). Figure 3.14 shows the comparison between the global and the local attention of Luong, Pham, et al. (2015b).

We can see that the global attention calculates the weights using all the input vectors (h) while in the right part of the figure, the local attention calculates the weights with only a window of input vectors plus a position information for the current step p_t.

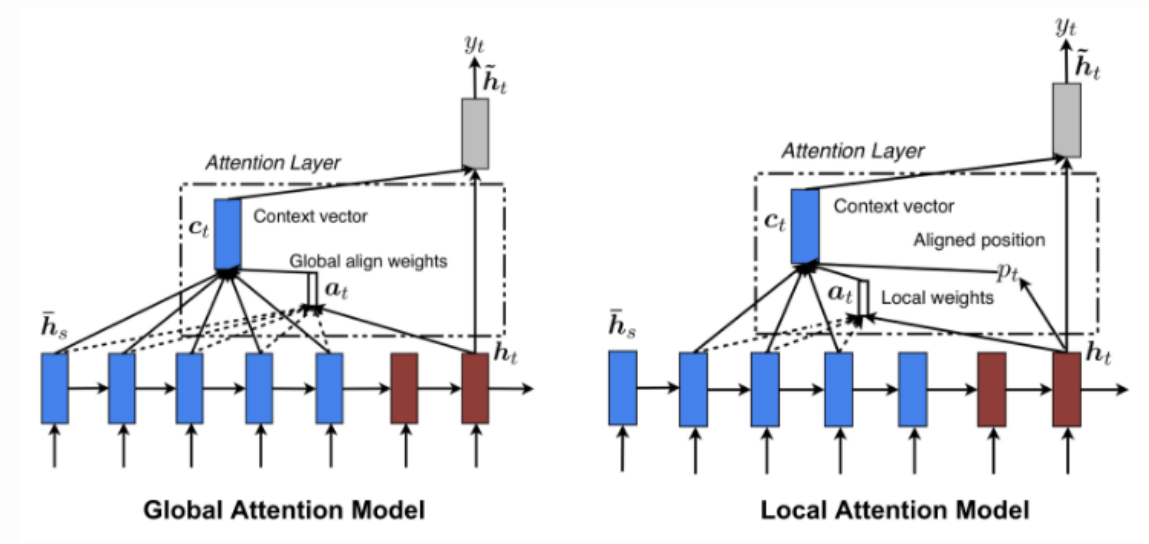


Figure 3.14 – Global vs local attention. (source: Luong et al. (2015).) Note h in this figure is the input for the attention layer which is v in our manuscript.

3.4.2 One head: Scaled Dot-Product Attention

In fact we can generalize attention with key-value pairs, given a sequence a key-value pairs $(k_i, v_i), i \in [1, N]$, the attention is:

$$\begin{aligned} \text{attention}((K,V),q) &= \sum_{i=1}^{N} \alpha_i v_i = \sum_{i=1}^{N} \text{softmax}(s(k_i,q))v_i \\ &= \sum_{i=1}^{N} \frac{\exp(s(k_i,q))}{\sum_j \exp(s(k_j,q))} v_i \end{aligned} \tag{3.55}$$

where s is the score function. If $K = V$, this equals to Equation 3.53 with Equation 3.48. Based on this idea, Vaswani et al. (2017) introduce a scaled dot-product based key-value **self-attention**.

Given an input sequence $X = [x_1, x_2, ...x_N] \in \mathbb{R}^{N*d_{input}}$ and the output dimension d_{output}, the query Q, key K and value V matrices are first obtained by three linear transformations with different learnable parameters W_Q, W_K and Q_v:

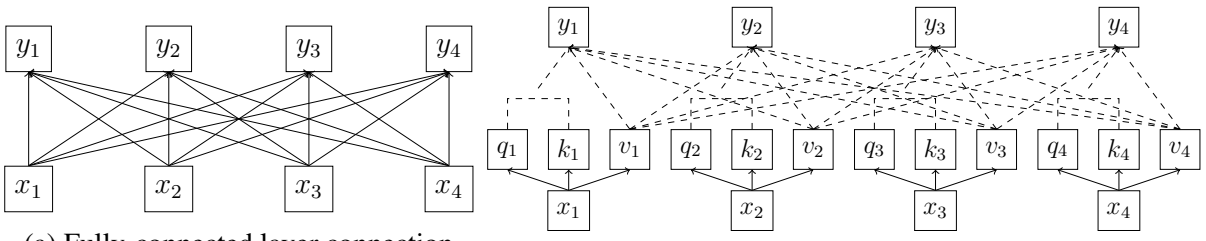
$$\begin{aligned} Q &= XW_Q \in \mathbb{R}^{N*d_m} \\ K &= XW_K \in \mathbb{R}^{N*d_m} \\ V &= XW_V \in \mathbb{R}^{N*d_{output}} \end{aligned} \tag{3.56}$$

where $W_Q \in \mathbb{R}^{d_{input}*d_m}$, $W_K \in \mathbb{R}^{d_{input}*d_m}$ and $Q_v \in \mathbb{R}^{d_{input}*d_{output}}$. d_m is the intermediate dimension size for queries and keys. The final attention has a scaled dot-product as the score function:

$$\text{attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_m}})V \in \mathbb{R}^{N*d_{output}} \tag{3.57}$$

where softmax is a column-wise function.

Intuitively, this self-attention can be considered as a fully-connected network with dynamic weight matrices. Both have the capability of memorizing long dependencies (Compared to RctNN and CNN). In a fully-connected layer, all inputs are connected to each output with learnable weights. One input x_i is connected or contributes to one output $y_j, j \in [1, J]$ with $W_{i,j}x_i + b_j$ where $W_{i,j}$ reveals actually the transformation direction. The input x_i is connected to the output y_j by a weight $W_{i,j}$. Once we have set the output size J we can no longer modify it. In the self-attention, one input x_i is first connected to all other inputs x_j with $i, j \in [1, N]$, by QK^T, and then multiplied by V to obtain the output whose length is always the same as the input length N. Thus we consider that the process is "dynamic" and the self-attention can handle variable length input. This is illustrated in Figure 3.15.



(a) Fully-connected layer connection

(b) Self-attention layer connection

Figure 3.15 – Comparison between a fully-connected layer and a self-attention layer of Vaswani et al. (2017). The solid lines represent a linear transformation with learnable parameters and the dashed lines mean a dot-product which does not require any parameters. We can see that the self-attention's is more dymanic since for any input with length N, the network use always the same parameters to generate a sequence of length N, while in a fully-connected layer, if we change the input size we will have to change the weight matrix.

In the work of Vaswani et al. (2017), one such attention as in Equation 3.57 is called "one head" which can be used as an alternative of a recurrent or convolutional neural network in any architecture.

3.4.3 Multi-Head for multiple semantic aspects

Like in a convolutional neural network, Vaswani et al., 2017 propose to add some "depth" to the attention layer to capture different aspect information: instead of performing a single attention function with one set of keys, values and queries, they linearly project the queries, keys and values h times with different, learned linear projections to d_m, d_m and d_{output} dimensions, respectively. Then, they perform the attention function(3.57) in parallel on each of these projected versions of queries, keys and values, yielding hd_{output} dimensional values in total:

$$\text{MultiHeadAtt}(Q, K, V) = \textbf{concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W^O \in \mathbb{R}^{N*d_{fo}} \tag{3.58}$$

where the i-th head, $\text{head}_i = \text{attention}(QW_i^Q, KW_i^K, VW_i^V)$ with $i \in [1, h]$. $W_i^Q \in \mathbb{R}^{d_{input}*d_m}$, $W_i^K \in \mathbb{R}^{d_{input}*d_m}$ and $W_i^V \in \mathbb{R}^{d_{input}*d_{output}}$ are the weight matrices in the i-th head. $W^O \in \mathbb{R}^{hd_{output}*d_{fo}}$ where d_{output} is the output dimension for one head and d_{fo} is the final output dimension for the multi-head attention.

The relation of one head and multi-head attention is shown in Figure 3.16.

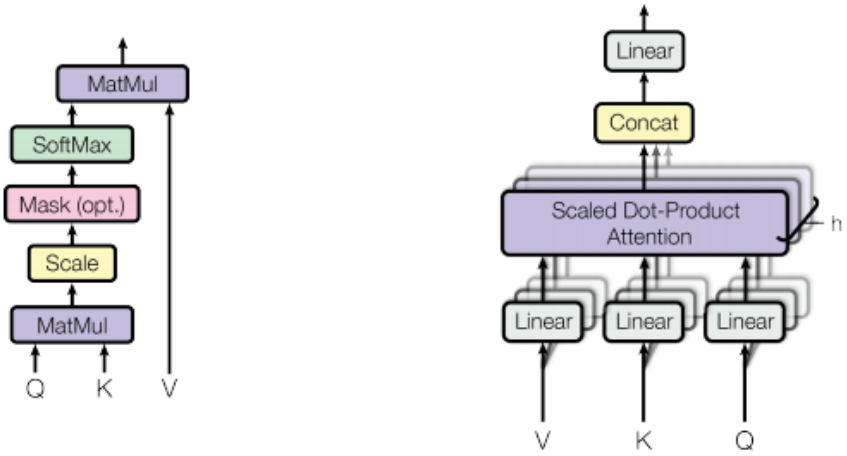


Figure 3.16 – The relation between one head (left) and multi-head attention (right). (Source: Vaswani et al. (2017))

3.4.4 Positional encodings for order distinction

Since the weight for one input x_i in the self-attention Equation 3.57 depends only on the linear transformation of q_i and k_i, ignoring completely the position of x_i in the input sequence, in case of using only self-attention layers, we usually add some order information via positional

encodings. For instance, we can use sine and cosine functions of different frequencies:

$$\begin{aligned} \text{PE}(pos, 2i) &= sin(\frac{pos}{10000^{2i/d_o}}) \\ \text{PE}(pos, 2i+1) &= cos(\frac{pos}{10000^{2i/d_o}}) \end{aligned} \tag{3.59}$$

where d_o is the dimension size of the input vector embeddings, for example, $d_o = d_{input}$ in our manuscript. pos represents the input position in the sequence. i is the dimension index of the positional encoding. Here, each dimension corresponds to a sinusoid. The periodicity allows the model to easily learn to attend by relative positions because for any fixed offset k, $PE(pos + k)$ can be represented as a linear function of $PE(pos)$.

3.4.5 Recent applications: Transformer, Opengpt, BERT, XLNet

Since 2017, the multi-head attention has been integrated in many general sequence modeling works which have advanced the state-of-the-art results (Vaswani et al., 2017; Devlin et al., 2018; Radford et al., 2019; Zhilin Yang et al., 2019).

The very first application is the ***transformer*** which is proposed in Vaswani et al. (2017) right after introducing the multi-head attention. The encoder contains 6 layers of multi-head attentions which combines 8 different heads and output 512 dimensional attention vectors. And they set d_m equal to d_{output}. Then it is followed by a normalization of $(x + \text{MultiHeadAtt}(x))$ where x is the input, and finally passed to a fully-connected layer with a second normalization of $(x + \text{Dense}(x))$ where x is the previous output of the multi-head attention layer. The decoder uses the similar architecture except that it inserts another multi-head attention sublayer to connect the encoder's output and masks the first multi-head attention layer. (See Figure 3.17)

Shortly after, it is extended by Z. Dai et al. (2019), proposing ***transformer-XL*** to remove the fixed-length context limit during the training of language models. In the same line, Radford et al. (2019) propose their language model ***GPT-2*** pre-trained with an objective of simply predicting the next word given the previous sequence with transformer units.

All these works train the model with whether unidirection or bidirection but in a separated manner, Devlin et al. (2018) introduce a pre-trained generic model for a wide range of NLP tasks with joint bidirection training, *Bidirectional Encoder Representations from Transformers* (***BERT***). A brief comparison between the training of these models is given in Figure 3.18.

Concerning the training objective, apart from the next sentence prediction in many general language models, they also add a *masked language model* (**masked LM**) training objective which predicts only some randomly masked tokens. This process is very similar to the *denoising*

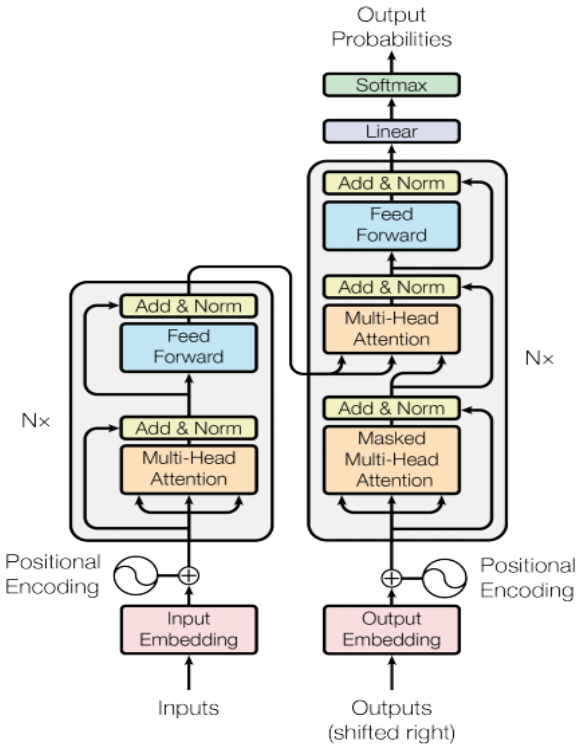


Figure 3.17 – Transformer model architecture. (Source: Vaswani et al. (2017))

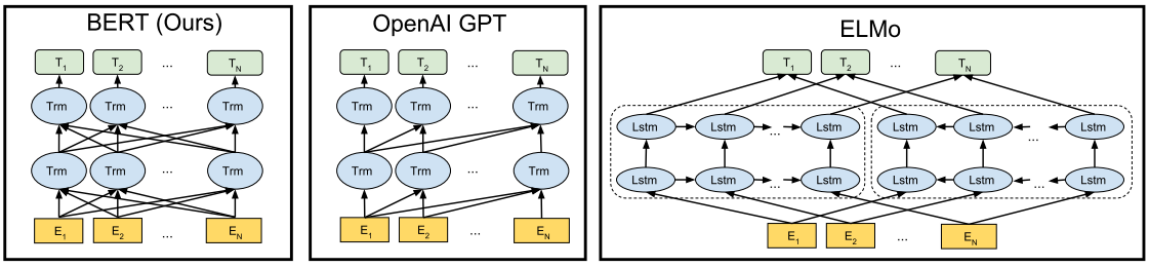


Figure 3.18 – Differences in pre-training model architectures. BERT uses a jointly-linked bidirectional Transformer. GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM. models. (Source: Devlin et al. (2018))

(Vincent et al., 2008) which will be discussed later, only in masked LM the prediction is only on the masked words rather than the entire input sequence. The authors state that by doing this the deep joint bidirectional model can be learned without allowing each word to indirectly "see itself" in a multi-layer context.

Compared to the previous models, BERT has achieved new state-of-the-art results on multiple GLUE tasks[7]:

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Figure 3.19 – Glue test results. Note that BERT_{BASE} and OpenGPT have comparable parameter number while BERT_{LARGE} is not yet publicly available. (Source: Devlin et al. (2018)) The data set details can be found in the original paper or the Glue official website.

Later, in June 2019, Zhilin Yang et al. (2019) argue that the denoising based training in BERT neglects dependency between the masked positions and suffers from a pretrain-finetune discrepancy. Based on the pros and cons of BERT and transformer-XL, they propose a generalized autoregressive pretraining method, ***XLNET***, which is a autoregressive model whose objective is to maximize the conditional probability $\prod_{k=1}^{T} P(x^k|x^1, x^2, ..., x^{k-1})$ just like ELMo or transformer-XL. In addition, they maximize the probability of all possible permutations of the factorization order, the idea is to simulate a joint bidirectional model to capture information from bidirectional contexts like BERT but without introducing noising-denoising processing. Let $\mathcal{Z}_t$ be the set of all possible permutations of the T-length input sequence, z_t and $\mathbf{z}_{<t}$ the t-th element and the first $t-1$ elements of a permutation $z \in \mathcal{Z}_t$, the training objective is to maximize the following averaged log likelihood:

$$\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_t}(\sum_{t=1}^{T} \log P(x_{\mathbf{z}_{<t}}|\mathbf{x}_{\mathbf{z}_{<t}})) \tag{3.60}$$

In our work, due to the heavy computation capacity requirement we have only chosen BERT as a baseline and an alternative input to the static embedding for our framework.

[7] https://gluebenchmark.com/leaderboard/

3.5 Contribution : a new dataset for monolingual phrase synonymy

For the phrase synonymy task, in addition to the dataset of Hazem and Morin (2017), we build a new dataset on the *Breast cancer* English corpus (BC-en). To extract the corpus, we crawl from a scientific website [8] by filtering all the publicly accessible articles and then search the keywords "breast cancer" for English articles. The final English corpus results from concatenating 168 articles and contains 26,716 sentences and 874,595 tokens. Our corpus crawler script is open source [9].

Concerning the the gold standard, we manually look for the most frequent terms of the corpus and then see if they have synonyms which exist simultaneously in the corpus and the MeSH 2018 thesaurus[10]. Finally the gold standard contains 108 synonymy phrase pairs.

3.6 Contribution : Tree-free recursive neural network

In previous sections, we have seen simple approaches which ignore the inner structure of tokens in a sequence. Regarding neural network based sequence modeling, there are various neural network architectures for inputs of variable length: convolutional, recurrent and self-attention. However, these architectures are proposed to encode natural sentence sequences which are often composed of more than 10 tokens, while our objective is to encode short sequences which are phrases of less than 10 tokens. Besides, RctNN can encode multiple tokens into one single vector (e.g., the hidden state for the last token) but this vector is biased by the last several words in a sequence due to the architecture property, and the self-attention only propose to encode a sequence to a sequence of the same length. Even if we can still choose the last or first token to represent the whole sequence, this vector is much biased by the last token as it is based on the linear combination of the key and query of the last step alone(See Figure 3.15). Although the experimental results in previous works are quite promising with this approach, the tasks are sentence-level classifications and the results are obtained by a supervised task-specific transfer learning process which differs from our scenario. With this being said, we propose a new architecture called *Tree-free recursive neural network* (TF-RNN) which is dedicated to encoding short sequences such as phrases. In this section, we first review the original recursive neural

[8] https://www.sciencedirect.com
[9] https://bitbucket.org/stevall/data-crawler.git
[10] https://meshb.nlm.nih.gov/search

network and then present our architecture model. Finally, we conduct a series of experiments on phrase synonymy and similarity, our results show that our proposal outperforms state-of-the-art results, and on the general domain corpora, the TF-RNN achieved even stronger results when integrating ELMo or BERT as an alternative input to the static word embedding.

3.6.1 Background: recursive neural network (RNN)

Recursive neural network (RNN) (Goller and Küchler, 1996) is a generalized version of the recurrent neural network (Elman, 1990) which always applies a left binary tree, where the first two leaves are combined to form a node, then the node is combined to the next leaf to form the next level node, etc. The recursive neural network encodes a sequence of word vectors along a tree structure, e.g. a parse tree, by recursively applying the weight matrices to each node association. This architecture has been successfully exploited in a variety of tasks, Socher, Bauer, et al. (2013) use an *untied weight* RNN for the constituent parsing where they use different weight matrices depending on the constituent syntactic category, P. Le and Zuidema (2014) collect the context information by adding an outer representation for each node. Their system is used in a dependency parsing task. Besides, various works (Socher, Perelygin, et al., 2013; Irsoy and Cardie, 2014; Paulus et al., 2014) apply the RNN to generate sentence level representation for sentiment analysis using some labelled data.

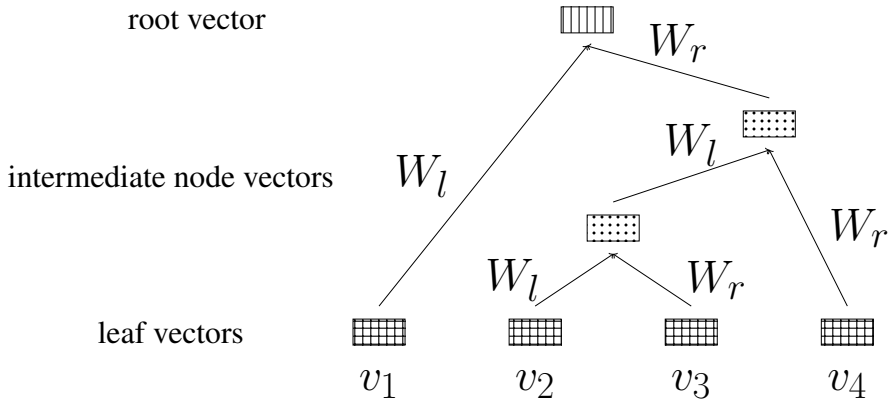


Figure 3.20 – Diagram of a recursive neural network.

Figure 3.20 shows an example of a sequence of length four. Suppose we have a parse tree, each input is a word vector $v_i \in \mathbb{R}^d$. The network applies a linear function with a weight matrix $W_l \in \mathbb{R}^{d*d}$ for each left node child and a weight matrix $W_r \in \mathbb{R}^{d*d}$ for each right node child in

the given tree. So for each non-leaf node η, the corresponding vector x_η is calculated as follows:

$$x_\eta = W_l v_{l(\eta)} + W_r v_{r(\eta)} + b \tag{3.61}$$

where $v_{l(\eta)}$ and $v_{r(\eta)}$ mean respectively the left and the right child vector of the node η.

The RNN is particularly interesting for us because short sequences or phrase association can be traced with the parse tree. By nature, the network distributes association weights for each element in a phrase. Therefore a phrase representation is not always biased by the last token like in the RctNN. However, the disadvantage of RNN in our scenario is the need of a tree structure because not only it is not always available in all languages, but it is also not possible to retrieve the context sentence for the parsing if we meet a new freely combined phrase that has never occurred in the corpus. The recurrent neural network or the LSTM does not need a tree structure but applies a universal left binary tree to all sequences, while the convolutional neural network with a kernel size of 2 can be considered as a specialised RNN where it adopts element-wise multiplication rather than matrix multiplication with only one layer by a pooling operation.

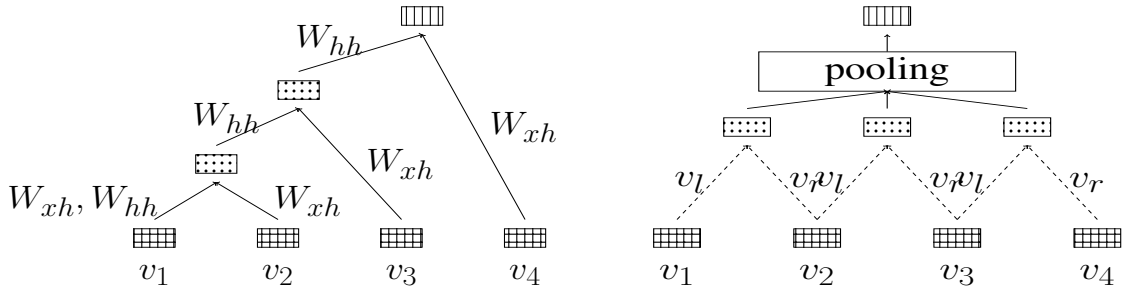


Figure 3.21 – Diagram of recurrent neural network (left) and 2 kernel sized convolutional neural network (right). For the purpose of clarity, we omit the output layer in the recurrent neural network.

Figure 3.21 shows how the recurrent and a 2 kernel sized convolutional neural network model for a sequence of 4 tokens. $W_{xh} \in \mathbb{R}^{h*d}$ and $W_{hh} \in \mathbb{R}^{h*h}$ are the parameters in a typical recurrent neural network where h is the hidden dimension, and for the convolutional network with a kernel size of 2, we can consider the convolution operation as two element-wise multiplications (dashed line in Figure 3.21) with a left multiplier $v_l \in \mathbb{R}^d$ and a right multiplier $v_r \in \mathbb{R}^d$. Stacking v_l and v_r forms the actual convolution kernel. The final vector is obtained by a pooling operation such as max or average. Note that the addition based approach (Liu et al., 2018) can be viewed as a specialised version of CNN where the values in v_l and v_r are fixed to

be one and the pooling is an averaging process.

3.6.2 Tree-free recursive neural network (TF-RNN)

In order to encode phrases of variable length without tree structures into a single fixed-length vector, we propose a new network called ***tree-free recursive neural network*** (TF-RNN). We consider it as a variant of the original recursive neural network because the basic idea is still to associate each token following a bottom-up structure. This structure is required as input of the original recursive neural network while in the TF-RNN, we eliminate this requirement by recursively splitting each node into a left and a right semantic part, then associating the left part with its right neighbour and the right part with its left neighbour. This is motivated by our hypothesis that the semantics of a pair of words could be retrieved by combining their meaning with some position-specific weights, and consequently the semantics of a sequence of words could be retrieved by recursively combining the semantics of each word pair. In fact, by doing this we create a pseudo binary tree structure where we associate each adjacent node pair without parsing it twice. This kind of structure can be seen as an approximation of a generalized sentence syntax as each language unit is directly associated with its adjacent neighbours and hierarchically associated with other units eventually yielding the overall semantics of all the units.

Let $[v_1^0, v_2^0, v_3^0, ..., v_n^0]$ with $v_i^0 \in \mathbb{R}^d$ be the input word vector sequence with n words, the TF-RNN outputs a single fixed-length vector $v_o \in \mathbb{R}^p$ by following steps:

$$\begin{aligned} v_{i,l}^j &= \tanh\Big(W_l v_i^j + b_l\Big) \\ v_{i-1,r}^j &= \tanh\Big(W_r v_{i-1}^j + b_r\Big) \\ v_i^{j+1} &= \tanh\Big(v_{i,l}^j + v_{i-1,r}^j\Big) \\ &... \\ v_o &= \tanh(U v_0^n + b) \end{aligned} \tag{3.62}$$

where j indicates the pseudo-tree structure layer level. A phrase with n word components will have n levels in such a structure. $W_l \in \mathbb{R}^{d*d}$ and $W_r \in \mathbb{R}^{d*d}$ respectively represent the left and right weight matrices for the extraction of the word semantics; $b_l \in \mathbb{R}^d$ and $b_r \in \mathbb{R}^d$ are the corresponding bias vectors. A node vector in the level $j+1$, v_i^{j+1} is calculated in terms of a pair of adjacent node vectors from the previous level j. Once we reach the final level n, the final output vector v_o can be calculated by a linear layer on top with $U \in \mathbb{R}^{p*d}$ and $b \in \mathbb{R}^p$ as

its parameters. A non-linear activation function is applied after each operation. An example of a sequence of three words ($n = 3$) is illustrated in Figure 3.22.

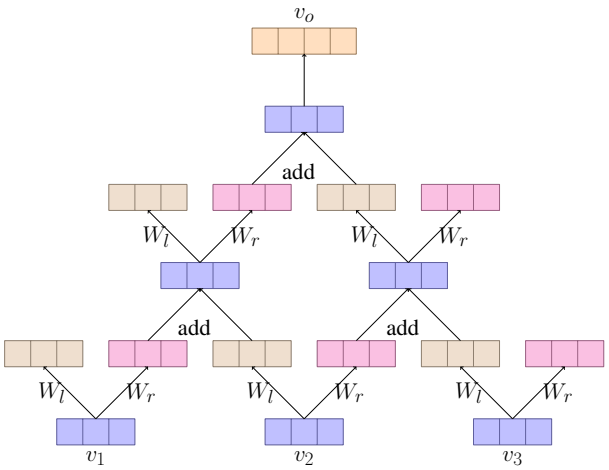


Figure 3.22 – Diagram of the tree-free recursive neural network (TF-RNN).

Clarification about the boundary

With a sequence of n words, each layer $0 <= j < n$ will have $n - j$ vectors. At layer j, the leftmost W_l matrix (resp. rightmost W_r matrix) is not included in the calculation for level $j + 1$. For single words we apply zero-padding to the left and right, so single-words' output is actually the sum of the left and right matrices.

Complexity

We compare the complexity of different neural network layers which can encode sequences of variable length, the RctNN and self-attention encode the input sequence to another sequence of equal length while our proposal TF-RNN and CNN with padding and pooling (right part of Figure 3.21) encode to one fixed-length vector.

We compare mainly two criteria for one layer of each architecture, the first is the computational complexity which represents how many weight parameters are involved in the linear transformation; the second is the maximum dependency length which is the length of the paths forward and backward signals have to traverse in the network. This is critical for learning long-range dependencies in many sequence transduction tasks. We show the complexity comparison in Table 3.1.

For the computational complexity, the RctNN has n times linear transformations with matrices in $\mathbb{R}^{d*d}$, while the CNN is more expensive than RctNN by a factor of the kernel width k which can be seen as having k times the weight matrices in a RctNN. The self-attention is faster for most of cases since usually $n < d$. Our proposal seems to be the most expensive since the

	RctNN	CNN	Self-Att	TF-RNN
Computational complexity	$O(n \cdot d^2)$	$O(k \cdot n \cdot d^2)$	$O(n^2 \cdot d)$	$O(\frac{n(n-1)}{2} \cdot d^2)$
Dependency length	$O(n)$	$O(\frac{n}{k})$	$O(1)$	$O(\frac{n}{2})$

Table 3.1 – Comparison of complexity. n is the input sequence length, d means the model dimension where we assume $d = d_{input} = d_{hidden} = d_{output}$ for simplifying the comparison. k is the kernel width for CNN.

complexity is quadratic in terms of input length and model dimension, however, keep in mind that our objective is to encode phrases which are most of the time n-gram with $n \in [1, 5]$. For unigrams and bigrams our encoder is less complex than RctNN and CNN. Besides, our encoder is a one-layer architecture compared to the self-attention which has a "depth" of 8 heads in the Transformer-base architecture.

The shorter the dependency length is, the easier the model memorizes long dependencies. Because the self-attention is a dynamic fully-connected layer, it traces each input position with one linear transformation. As for our proposal, we can see that it is linearly related to the input length, yet again since our inputs are mostly short sequences, this is not considered as problematic in our scenario.

3.6.3 Evaluation

We have conducted a set of experiments on the phrase synonymy and similarity tasks. **Baselines** approaches are pre-trained models and **context** means that we train the corresponding phrase encoder with an encoder-decoder architecture which will be detailed in the next chapter. The BERT and ELMo inputs can be categorized as **contextualized embeddings** considering their context training properties. **Static** means either the 400 dimensional static word embedding vectors obtained from concatenating the pre-trained *fastText*[11] vectors and vectors trained on small specialised domain corpora for the phrase synonymy task (See Section 2.4 for more details), or the 64-dimensional static word embedding vectors provided by the *Semeval 2017* [12] dataset for the phrase similarity task. **Skip-gram-ext** is the extended Skip-gram presented in Section 3.2.2 (Artetxe, Labaka and Agirre, 2018b). Finally, we have three types of input embedding:

- **Skip-gram-ext**. 300 dimensional embeddings trained on the specialized domain corpora

[11] `https://fasttext.cc/`
[12] `http://alt.qcri.org/semeval2017/task2/index.php?id=data-and-tools`

where N-gram phrases are considered as one single unit in the embedding look up table.

- **Static**. The 400 dimensional static word embedding vectors obtained from concatenating the pre-trained *fastText* vectors and vectors trained on small specialized domain corpora.
- **BERT** or **ELMo**. Pre-trained contextualized embeddings with feature-based usage setting.

In addition, we have explored three different methods to pool a single vector representation from a vector sequence:

- **mean**. The addition based approach can be applied to both the static and the contextualized input embeddings.
- **reduce**. Use an output at one specific step of a sequence to represent the whole input sequence for the contextualized embeddings. For *ELMo* we use the last step as in many standard sequence model while for *BERT* we use the first step which is the special token "[CLS]" as it has been used for the classification tasks in Devlin et al. (2018).
- **ELMo-Peters et al. (2018)**. The original pooling method proposed in the work of *ELMo*. It concatenates the first and the last step token embeddings to represent a sequence. Note that for single-words, we simply duplicate the only embedding and concatenate the two identical vectors.

To compare our proposed TF-RNN encoder with other architectures, we also implemented several neural networks which do not structured input: **RecurrentNN, CNN** and **Transformer** cell. To conclude, we have four types of phrase encoder:

- **CNN** (0.4M parameters). The CNN has a kernel size of 2 and a zero-padding to the left and right of one phrase so that even single-word phrases can be encoded.
- **RctNN** (0.5M parameters). A regular recurrent neural network which always applies a left binary tree to generate the final vector.
- **Transformer**. (5M parameters). In order to be comparable with other architectures in parameter number, we use a small Transformer encoder with 4 layers and 4 heads, the hidden dimension size is 2 times the model dimension.
- **TF-RNN** (0.5M parameters). Our proposed phrase encoder presented above.

Phrase synonymy

Each dataset of the phrase synonymy task contains phrases of variable lengths and for each phrase, the reference phrase can be different in length, for example, “wind turbine” and “windmill”. The results are shown in Table 3.2.

	Method	Synonymy dataset		
		WE-fr	WE-en	BC-en
Baselines	Skip-gram-ext	<0.5	<0.5	23.30
	Static-mean	5.29	12.19	39.65
	BERT-reduce	4.07	10.44	26.04
	BERT-mean	4.49	16.59	36.58
	ELMo-reduce	1.54	4.09	26.23
	ELMo-mean	7.37	5.20	29.27
	ELMo-Peters et al. (2018)	8.97	9.60	28.28
Context	Static-CNN	7.42	15.71	35.75
	Static-RctNN	12.89	20.53	42.60
	Static-Transformer	4.62	15.82	35.90
	Static-TF-RNN	**15.06**	**33.47**	**44.84**

Table 3.2 – Overall MAP comparison for the phrase synonymy task. We have introduced the data set for BC-en in Section 3.5. Details for the WE corpus can be found in Appendix A.1.2 and the gold standard for WE is in Appendix A.3.1.

Phrase similarity

The phrase similarity task dataset also contains phrase comparison pairs of variable lengths such as “Harry Potter” and “wizard”, “window blind” and “curtain”. The results are shown in Table 3.3.

Discussion

Our approach with concatenated static word embeddings and the TF-RNN as phrase encoder has the best results in phrase synonymy task. While the *Transformer* encoder has achieved the best result for *Semeval2013* and the second best result for *Semeval2017* phrase similarity task. The TF-RNN has managed to obtain the third best result for the phrase similarity task on *Semeval2017*. Given that the *Semeval2013* dataset does not provide any textual data and the model is trained on the textual corpus of *Semeval2017*, the results on *Semeval2013* for the context prediction approaches are biased by the data availability. Moreover, even compared to the approaches with contextualized embedding input, for the most of the time the context prediction approaches (last four lines) have better results on the datasets that provide a textual

	Method	Similarity dataset	
		Semeval2013†	Semeval2017
Baselines	Skip-gram-ext	0.378	**76.827**
	Static-mean	26.910	38.843
	BERT-reduce	0.754	12.735
	BERT-mean	19.482	36.378
	ELMo-reduce	35.112	37.968
	ELMo-mean	37.991	36.207
	ELMo-Peters et al. (2018)	36.233	31.420
Context	Static-CNN	*(29.890)*	42.245
	Static-RctNN	*(21.720)*	42.961
	Static-Transformer	***(39.524)***	49.324
	Static-TF-RNN	*(22.003)*	44.382

Table 3.3 – Overall comparison of correlation score for the similarity task. The Semeval correlation score is the harmonic mean of Pearson and Spearman scores. A † indicates that the corresponding corpus for training a neural network is not available. In the case of the context based approaches, we use the network trained on the Semeval2017 corpus, as it is also a general domain corpus. Details of the data set can be found in Appendix A.2.2 and A.3.2

.

corpus to train the model. Although the contextualized embedding models capture the inner relation between each component word in a phrase, it cannot exploit the context information of the phrase during the test phase or if the phrase is out of the training corpus, while the encoder-decoder training based approach memorizes and generalizes the context information of different phrases in the training corpus (See the next chapter). This confirms again the hypothesis of Harris (1954) and that the context information of a phrase is meaningful for learning the phrase representation. Moreover, if we compare different phrase encoders, our proposed *TF-RNN* outperforms the existing neural networks (even the Transformer encoder with significantly more parameters) on the synonymy task on every dataset and obtains comparable results on the similarity tasks. Therefore we believe that carefully representing the phrase following a relevant syntactical structure can generate better vector representations.

Among the non encoder-decoder training based approaches (the first seven lines), first the extended Skip-gram works very poorly for the phrase synonymy task. This is because many phrases during the inference are freely combined so they may not appear in the training corpus. As a consequence, these phrases do not have any representation in the look-up table. This phenomenon can also be observed on *Semeval2013* similarity task. However, it performs sur-

prisingly well for the similarity task on *Semeval2017*. The reason probably lies in the fact that *Semeval2017* has a large training corpus and all the phrases in our test are presented in the training corpus. We also notice that the contextualized embeddings (from the second to the fifth lines) are not better than the static embeddings for short sequence tasks. In fact the static embeddings hold the best results on the BC dataset. For the contextualized embedding models, it seems that the mean of each output vector works better for BERT and each of the three pooling method has its own advantage for ELMo. When comparing the BERT and ELMo models with mean representation, we can see that the BERT model has relatively decent results on the English synonymy datasets while the ELMo model is more effective on the French dataset and the similarity task. Our explanation is that, as the BERT model is a multilingual model mixed with 104 languages, it is not surprising that the model is biased by the English training corpus. By contrast, the ELMo French model is a separate model trained only on French data. For the similarity task, the ELMo model largely outperforms the BERT model on the *Semeval2013* dataset although the two models have similar results on the *Semeval2017* dataset.

In addition, we provide a qualitative evaluation on the phrase analogy task using the stactic word embeddings. The phrase analogy task is identical to the word analogy task, we only replace words by longer linguistic units. We try to address the question *a to a' is like b to ?*, formally written as $a : a' :: b : b'$, as in the famous example of "man to woman is like king to queen", and "king - man + woman" results in a vector very close to "queen" (Mikolov, Sutskever, et al., 2013). Since we have no available phrase analogy reference, we show 3 representative samples found in the WE corpus to illustrate the properties of the proposed model compared to the state-of-the-art approaches by exploratory analysis.

The results are shown in Table 3.4. Clearly, our proposed model achieve better results compared to addition based representations. Moreover, our model manages to reduce the penalty inherited from the word embedding problem reported in other works that $b - a + a'$ results basically in a vector close to b or a' (Schluter, 2018). This phenomenon can been observed more easily from the results of the addition based approach, the results are always phrases similar to b or a'. However our model shows relatively strong relatedness to all the three inputs. If we look at the results from the additive approach, "wind power tank - wind power capacity + hydrogen storage" basically leads to a vector close to "hydrogen storage", thus missing the storage device meaning that our model is able to retrieve. The same happens to "marine animal - marine mammal + marine epifauna" which leads to "epifauna" while we are able to retrieve "marine fish". We also observe that the model is able to capture the key information in the phrases which is not always the syntactic head, for example it is "safety" not "glass" that should appear in the

TF-RNN	**ADDITION**
wind power capacity : hydrogen storage :: wind power tank : ?	
hydrogen storage device	storage of hydrogen
gas storage tank	hydrogen storage system
hydrogen storage system	**hydrogen storage device**
hydrogen system	gas storage tank
hydrogen solution	hydrogen
marine mammal : marine epifauna :: marine animal : ?	
marine fish	epifauna
marine environment	marine
marine site	non marine
conservation of marine biodiversity	**marine fish**
marine zone	marine environment
safety equipment : safety standard :: safety glass : ?	
safety concept	glass fibre
safety requirement	make of glass
carbon glass	carbon glass
make of glass	safety
glass fibre	occupational safety

Table 3.4 – Top 5 phrases similar to $b - a + a'$

phrase close to "safety glass - safety equipment + safety standard", the addition model vector is clearly overwhelmed by the meaning of "glass". Overall, phrase representation of TF-RNN improves the rank for highly related phrases for each analogy.

3.7 Synthesis

Modeling sequence representation has been a challenging task. In this chapter we first study the relatively simple approaches which **do not require parameter training**. Despite of the simplicity, the **addition** approach has always a fairly acceptable performance on many NLP tasks including our synonymy and similarity tasks. One drawback of these approaches is that they all ignore the inner relation between phrase components.

Another simple approach to obtain sequence representation is to treat the sequence as one single token. By doing this we can easily apply single-word representation learning approaches such as CBOW or Skip-gram. In this line, the work of Artetxe, Labaka and Agirre (2018b) proposes an **extended skip-gram** in order to encode phrases. Typically, this approach cannot handle freely combined phrases that have not been "seen" during training.

In order to take the inner relation of phrase components into account while still being able to generate a representation for freely combined phrases, we can rely on models that distribute different weights to each token with regard to their position and association manner of the phrase. This can be achieved by several neural networks such as the **recurrent neural network (RctNN)**, the **convolutional neural network (CNN)** and the popular variants of the recurrent neural network, e.g, **Long short-term memory (LSTM)** and **gated recurrent unit (GRU)**. One of the successful applications is the **embedding from language models (ELMo)** which can be seen as a substitution of the static word embeddings such as CBOW and Skip-gram.

More recently, **multi-head attention** unit based architectures have made significant progress on a wide range of general NLP tasks. As nearly all of these models incorporate the **Transformer** (Vaswani et al., 2017) layer which contains several multi-head attention layers, we categorize them as transformers, including the original **Transformer**, **BERT, Open-gpt** and **XLNET** which is the latest application to date (June 2019). Note that these models encode sequences of variable lengths into sequences of the same length, while we would like to encode sequences of variable lengths into one single fixed-length vector. Although we can use the representation at one particular step, we assume it is biased by the input at that step by analysing the computation.

Finally, we propose a new architecture based on the **Recursive neural network (RNN)**

which is often used for constituent parsing with a tree structure, and remove the dependency of a tree structure since it is not a trivial resource for us. We call it the **Tree-free recursive neural network (TF-RNN)**. The architecture fits quite well our objective of encoding phrases which are actually short sequences. Our experimental results on specialized-domain corpora have outperformed state-of-the-art systems. Moreover, this architecture with BERT or ELMo as input layer has shown promising results on our experiments on general-domain corpora.

CHAPTER 4

UNSUPERVISED TRAINING

In the previous chapter, we presented plenty of neural networks for encoding sequences. But we have not seen how these networks can be trained without supervised information. Recall that in our scenario, we would like to learn phrase representations using only textual information. In this chapter we will cover two classes of unsupervised training strategies: the Siamese network and the encoder-decoder language modeling. Next we discuss how we can exploit the pre-trained models which can be split in two kinds: the feature extraction based approach and fine-tuning based approach. Finally we propose a new unsupervised training objective for learning phrase representations. On the phrase synonymy and similarity tasks, the results on comparison of different training objectives show the effectiveness of our proposal.

4.1 Siamese networks

Siamese networks were introduced by Bromley et al. (1993) and consists of two identical networks joined at their output with shared parameters θ. This kind of network can be learned by back-propagating the distance between the two outputs. The training objective is to minimize the encoding distance of the two branch networks or maximize the similarity, e.g., the cosine similarity.

$$\begin{aligned} &\underset{\theta}{\operatorname{argmin}} \, ||\text{BN}(x_1|\theta) - \text{BN}(x_2|\theta)|| \\ &\underset{\theta}{\operatorname{argmax}} \, cos(\text{BN}(x_1|\theta), \text{BN}(x_2|\theta)) \end{aligned} \tag{4.1}$$

where BN means one branch network forward pass. This regression model can train a network to capture common features of similar inputs. It has been successfully applied to many computer vision works (Koch et al., 2015; Zagoruyko and Komodakis, 2015; Liwei Wang et al., 2016; Martin et al., 2017). In NLP, Kenter et al. (2016) exploit it in the CBOW model (Mikolov, Sutskever, et al., 2013) to get word embeddings which have better performance for sentence representations, while Das et al. (2016) use a deep convolutional Siamese network to retrieve similar questions. This kind of network is widely used in the computer vision domain. The

basic idea is shown in Figure 4.1:

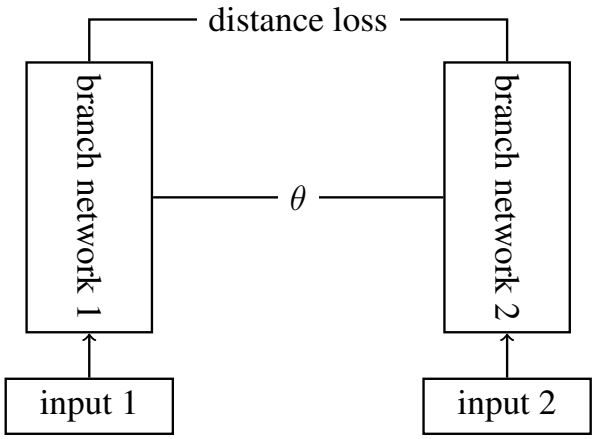


Figure 4.1 – A Siamese network.

4.1.1 Pseudo-Siamese network

A pseudo-Siamese network is a network with two branches that don't share the same parameters (Zagoruyko and Komodakis, 2015; Liwei Wang et al., 2016). Our baseline model with the pseudo-Siamese network tries to learn the phrase representation by minimizing the distance between the phrase and the context (the sentence containing the phrase), this is also inspired by the distributional hypothesis (Harris, 1954), i.e., *words in similar contexts tend to have similar meanings*. In Figure 4.1, each branch has its own parameters, denoted by θ_1 and θ_2.

$$\begin{aligned} &\underset{\theta_1,\theta_2}{\operatorname{argmin}} \, ||\mathrm{BN}_1(x_1|\theta_1) - \mathrm{BN}_2(x_2|\theta_2)|| \\ &\underset{\theta_1,\theta_2}{\operatorname{argmax}} \, cos(\mathrm{BN}_1(x_1|\theta_1), \mathrm{BN}_2(x_2|\theta_2)) \end{aligned} \tag{4.2}$$

In regard to our work, it seems quite suitable to apply a pseudo-Siamese network to train our phrase encoder. Following the distributional hypothesis, it is logical to assume that a phrase representation is close to its surrounding contexts. Therefore we use a phrase encoder as a branch network and a context encoder as the other branch in our pseudo-Siamese network. We will present the results later in this chapter.

4.2 Encoder-decoder architecture

The encoder-decoder architecture, also known as *seq2seq*, is a model that aims to map a fixed length input with a fixed length output where the length of the input and output may differ. Introduced by Sutskever et al. (2014), the encoder-decoder architecture has been successfully applied to an extensive range of fields such as automatic summary and question anwsering. And in NLP, it is mainly used in *neural machine translation*, where the encoder network encodes a source language sentence and the decoder network decodes it into a target language sentence with a potentially different length. The overview of this architecture is shown in Figure 4.2.

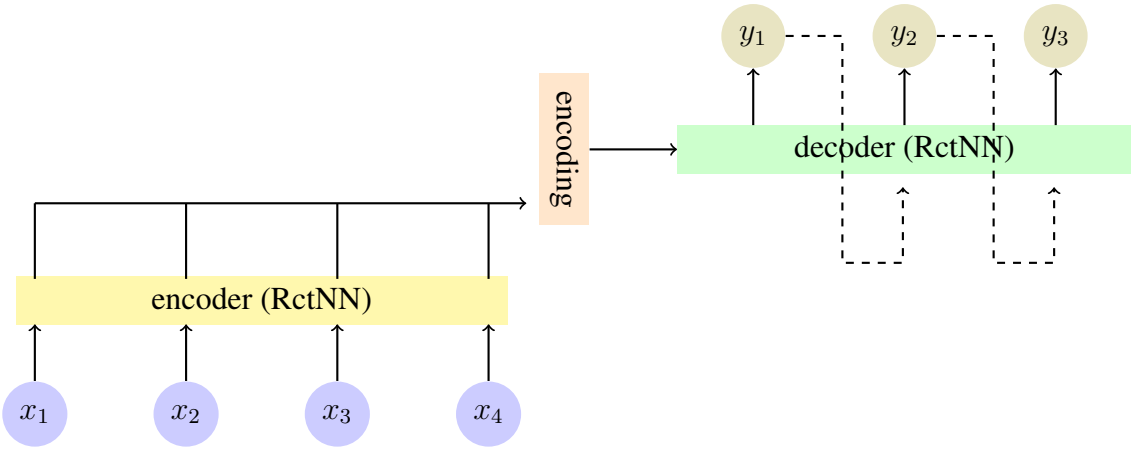


Figure 4.2 – Diagram of a general encoder-decoder architecture. A common framework uses RctNN or its variants as the encoder and the decoder.

Recall that the attention mechanism described in Section 3.4.1 can be optionally integrated for the encoder in order to tackle long dependency problems (Bahdanau et al., 2014; Luong, Pham, et al., 2015b).

As for the encoder and the decoder, the most common architecture is the RctNN and its variants such as the LSTM and the GRU. Usually, we stack multiple layers in order to capture more complex relations and use bidirectional layers for the encoder so that both left-to-right and right-to-left language dependencies affect the sequence encoding, as presented in Figure 4.3.

The stacked RctNN can be concluded as:

$$h_t^l = f(U^l h_{t-1}^l + W^l h_t^{l-1} + b^l) \tag{4.3}$$

where l means the l-th layer and t is the time step. The hidden states for the first layer h_t^0 equals to the input x_t. Concerning the bidirectional RctNN, the hidden states of each direction are

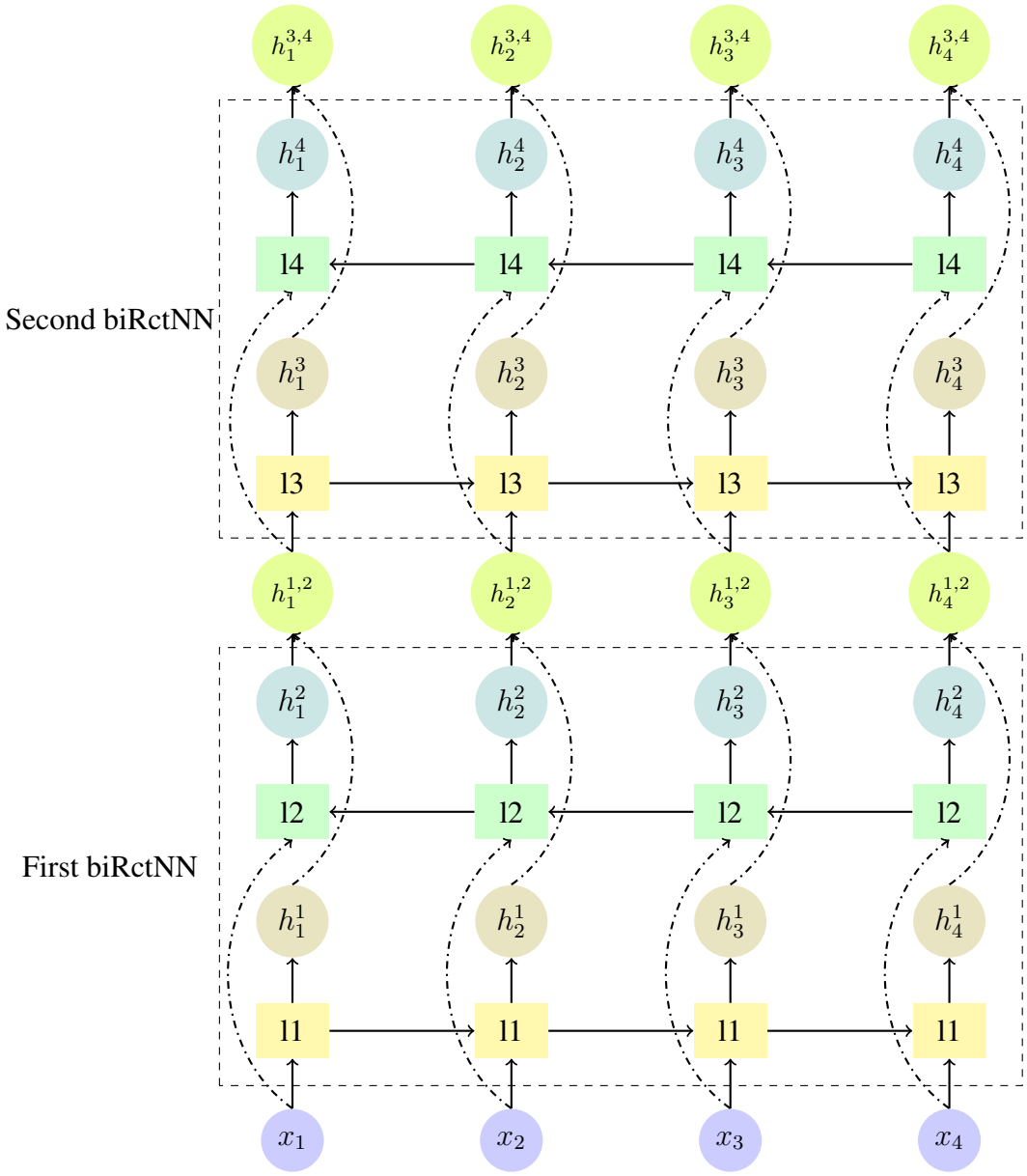


Figure 4.3 – Diagram of a stacked bidirectional encoder architecture. Solid lines represent the linear transformation and dash-dotted lines the concatenation.

concatenated to generate a new double-dimensional hidden vector. There are two sub-layers in one bidirectional RctNN layer, suppose that the first sub-layer is a left-to-right layer and the second sub-layer is a right-to-left layer, then for one bidirectional layer we have:

$$\begin{aligned} h_t^1 &= f(U^1 h_{t-1}^1 + W^1 x_t + b^l) \\ h_t^2 &= f(U^2 h_{t+1}^2 + W^2 x_t + b^2) \\ h_t^{1,2} &= h_t^1 \oplus h_t^2 \end{aligned} \tag{4.4}$$

where $\oplus$ means the concatenation and $h_t^{1,2}$ implies the concatenated vector of h_t^1 and h_t^2, x_t is the current input at time step t. Note that in a stacked bidirectional RctNN scenario, $x_t = h_t$.

4.3 Training objectives

In this section, we discuss the training objectives to train a sequence model. These training objectives are mainly used in an encoder-decoder or autoencoder architecture which can be seen as a special case of an encoder-decoder architecture where the output has the same length and is in the same space as the input.

4.3.1 Autoregression: Next word prediction

The most common way of training a sequence model in an unsupervised manner is to use an autoregressive training objective (A. M. Dai and Quoc V Le, 2015; Peters et al., 2018; Radford et al., 2019): distributing the probability for the next or previous word given a sequence of words $[x^1, x^2, ...x^T]$. The training objective is to maximize the product of the conditional probabilities at each time step t:

$$\begin{aligned} &\arg\max_x \prod_{k=1}^{T} P(x^k | x^1, x^2, ..., x^{k-1}) \text{ In a forward model} \\ &\arg\max_y \prod_{k=1}^{T} P(x^k | x^T, x^{T-1}, ..., x^{k+1}) \text{ In a backward model} \end{aligned} \tag{4.5}$$

This objective allows us to train a sequence model without supervised downstream task data. All it needs is the text. The decoder, at the inference time, can have two different prediction behaviours. Let's see this a little bit more in detail.

Greedy Search

Greedy search is fairly straightforward: at each step of the prediction output, the model

selects the best candidate according to the current probability distribution. In other words, for the i-th output, the candidate with the highest probability, $\hat{y}^i$, will be stacked in the output sequence $[\hat{y}^1, \hat{y}^2, ..., \hat{y}^{i-1}, \hat{y}^i]$.

Beam Search

The output sequence of the greedy search algorithm is remarkably impacted if a bad candidate is selected at the earlier positions, because once the candidate is selected, it is fixed in the output sequence and all the next tokens are predicted according to the previous predictions which contain a false one. The common solution to this problem, instead of considering only one candidate sequence at each step, considers several alternative sequences that equals to the *beam width*, b. Note that these sequences should have the top b probabilities. Consequently, for the step i, we need to stock b sequences:

$$[\hat{y}_1^1, \hat{y}_1^2, ..., \hat{y}_1^{i-1}, \hat{y}_1^i], [\hat{y}_2^1, \hat{y}_2^2, ..., \hat{y}_2^{i-1}, \hat{y}_2^i], ..., [\hat{y}_b^1, \hat{y}_b^2, ..., \hat{y}_b^{i-1}, \hat{y}_b^i]$$

where $\hat{y}_k^i$ is the prediction for the i-th step of the k-th candidate sequence. Apparently, the greedy search is a special case of the beam search where the *beam width* equals to 1. Usually, beam search improves the prediction quality but also leads to an increase of the time and space complexity.

In practice, since the sequence length is often very long, the factorization of each probability tends towards zero, leading to calculation instability. Instead of calculating the factorization of the probabilities, we use the sum of the logarithm of each probability. This sum is then normalized by the sequence length:

$$\frac{1}{T^\alpha} \sum_{k=1}^{T} \log P(x^k | x^1, x^2, ..., x^{k-1}) \tag{4.6}$$

where α is a hyper parameter between 0 and 1 which determines the penalty for long sequences.

4.3.2 Denoising: Reconstructing input and Masked language modeling

Aside from the estimative training objective which predicts a probability distribution, another popular unsupervised training strategy is to reconstruct the input text from its noised or corrupted version. We call this kind of training objective *denoising*.

Undoubtedly, There are quite a lot of different ways to noise an input text. We present the state-of-the-art method which is applied in *BERT* (Devlin et al., 2018). Given an input text sequence, by the training data generator, a certain portion of tokens (0.15×0.8) are replaced by

a special symbol [MASK] and some other tokens (0.15×0.1) are replaced by a random token, and then the model is trained to recover the original text from the noised version.

Since an autoregressive language model is trained to encode a uni-directional context (either forward or backward), it is not effective at modeling deep bidirectional contexts. Because the input inevitably contains the gold tokens in a bidirectional context, the model will be biased since it has the gold information at each step. The denoising approach, however, can be applied in an bidirectional architecture as the input does not contain the masked tokens.

The noising approach is also popular in preparing the data for cross-lingual tasks following the idea of denoising autoencoders (Vincent et al., 2008), in order to truly learn the compositionality of the input text in a language independent manner, Artetxe, Labaka, Agirre and Cho (2018) alter the word order of the input sentence in a dual training system inspired by He et al. (2016). The denoising training objective, which only leverages the monolingual information, is to reconstruct the original version of the input sentence.

4.4 After the pre-training: feature based vs fine tuning

When the training of a model is finished, there are two ways of exploiting the pre-trained model. The first one, which is fairly straightforward, is simply to run the pre-trained model with the input and extract the output of some layers to form the features of the input. The second way consists in fine tuning the pre-trained model with a downstream task such as sequence labeling or classification. In this section we explain briefly these two strategies.

Feature based approach

The feature based strategy has a long history since the 1990s. The traditional co-occurrence count based method (Church and Hanks, 1990; Dagan et al., 1994; Niwa and Nitta, 1994; Bullinaria and J. P. Levy, 2007; P. D. Turney and Pantel, 2010) represents a word by a sparse co-occurrence vector and often applies the *pointwise mutual information* to associate the word and its context. Neural network based methods Mikolov, Sutskever, et al. (2013) and Pennington et al. (2014) represent a word by a dense embedding vector which has led to significant improvements in major NLP tasks. These word-level static vectors can be incorporated into other systems as the basic input units to generate higher level representations.

Fine tuning approach

Contextualized models (Peters et al., 2018; Devlin et al., 2018; Radford et al., 2019) are sequence-level representations with word-level granularity. In fact, they all exploit word-level representations as the basic input and output units. Once pre-trained, we can use these models

in a specific task by stacking for instance some top layers on them. The difference between the feature based and the fine-tuning based approach lies in whether we freeze the parameters of these pre-trained models or not when we incorporate them in a task-specific training framework.

Discussion

The feature based approach extracts the output of the pre-tained model and uses this output as static features of the input by omitting the gradients of the parameters, while the fine-tuning based approach updates its parameters during the back-propagation of the training. The advantage of the fine-tuning based approach is that the whole system can be readjusted to the task-specific training corpus but it is much more time and space consuming compared to the feature based approach. Moreover, according to Devlin et al. (2018), similar performance can be obtained (with -0.3 points in F1 CoNLL-2003 NER) using the same BERT model in the feature and the fine-tuning based settings. This is particularly interesting because fine tuning a large model with millions of parameters can be exceedingly long while updating only a few layers is much more efficient.

Pertaining to our work, we choose to apply the feature based approach for the efficiency and relatively comparable performance.

4.5 Contribution : Wrapped context modeling

In order to learn our phrase encoder in an unsupervised manner, we propose to exploit the encoder-decoder architecture with a *wrapped context prediction* training objective.

Similar to the *masked language modeling*, the *wrapped context prediction* is a denoising approach, however, instead of reconstructing the original input text, our proposal aims to predict the context of the input phrase. This seems to be similar to the *continuous bag-of-words* (CBOW) algorithm but CBOW does not take word order into account, while the decoder outputs each prediction based on previous predictions which is a sequential process.

Still, one disadvantage of predicting only the context is that the syntax of the output sequence is misguided by the missing phrase. Since most of the phrases are either nominal or verbal, we decide to use a single universal random vector to wrap all the tokens of a phrase to help the generator reconstruct a syntactically complete context during the system training.

As shown in the Figure 4.4, the *wrapped context prediction* training objective can be seen as a specialized version of a “conditional CBOW” or “context masked language modeling”. On one hand, each prediction is obtained by a conditional probability given the previous predictions. On the other hand, the input text is composed of only the phrase span, which means that

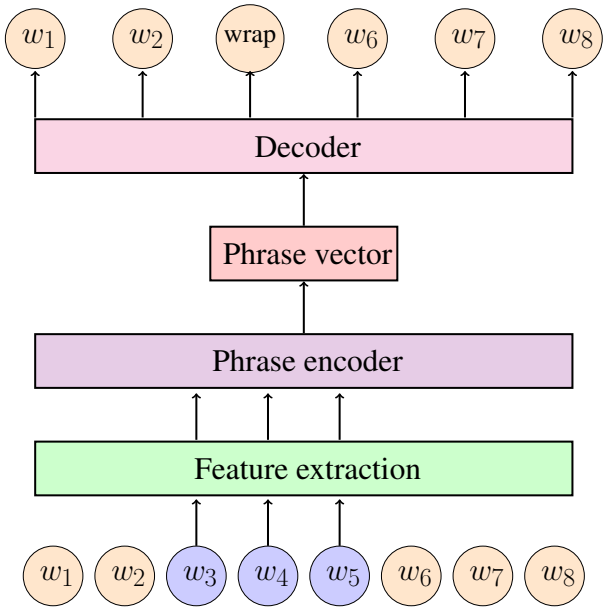


Figure 4.4 – Overview of the proposed framework, w_i is the i-th word in a sentence, *wrap* is the token represented by the randomly generated vector for filling the phrase blank when generating the context. In the example [w_3, w_4, w_5] is the phrase sequence.

the context is completely omitted.

Evaluation

For the sake of comparison, we have conducted the same monolingual experiments as in Section 3.6.3 with different training objective settings. The results are shown in Table 4.1. In order to prove the effectiveness of the proposed training objective, we evaluate two more models using two different training objectives with exactly the same experimental settings. The first one predicts all the sentence tokens, represented by "plain". The second one predicts only the context tokens around the phrase without the wrapped phrase token, represented by "context".

We can clearly see that the wrapped context training objective obtains consistently the best results compared to other possible objectives in our scenario. Although the context prediction strategy is fairly close, adding a wrapped token to replace the phrase allows the system to learn from a syntactically more complete sequence. Predicting all the tokens including the phrase is worse than the context prediction objective even if it predicts a syntactically complete sequence. The reason for this is possibly that predicting the phrase tokens makes the encoder over related to the specific phrase components rather than the generalized features across different but similar phrases, eventually it is difficult for the encoder to generate close vectors for these phrases.

Task	Training objectives		
	Plain	Context	Wrapped
WE-fr	9.40	13.35	**15.06**
WE-en	30.08	32.85	**33.47**
BC-en	39.48	41.49	**44.84**
Semeval2013†	16.759	21.376	**22.003**
Semeval2017	39.223	43.079	**44.382**

Table 4.1 – Results of our system with the TF-RNN encoder and static embeddings with different training objectives. The BC-en data set is introduced in Section 3.5. The WE data set information can be found in Appendix A.1.2 and A.3.1. The Semeval data set information is presented in Appendix A.2.2 and A.3.2.

In addition to the encoder-decoder architecture, we have also evaluated the pseudo-siamese network. Table 4.2 shows the results.

Task	Architecture	
	pseudo-siamese	encoder-decoder
WE-fr	3.84	**15.06**
WE-en	11.71	**33.47**
BC-en	32.18	**44.84**
Semeval2013†	0.345	**22.003**
Semeval2017	4.164	**44.382**

Table 4.2 – Comparison of the encoder-decoder framework with a pseudo-siamese network. The two systems use the TF-RNN as phrase encoder. The WE data set information can be found in Appendix A.1.2 and A.3.1. The Semeval data set information is presented in Appendix A.2.2 and A.3.2.

We can see that the pseudo-siamese network performs very poorly on all datasets, with extremely large drops on the similarity. It is somewhat unexpected for us because it seems that both frameworks follow the same distributional hypothesis (Harris, 1954). This may be due to the nature of the comparison tasks or the small size of our training samples. It perhaps explains why the encoder-decoder systems are becoming more popular in recent studies compared to others.

Given the fact that the siamese network perform poorly in our monolingual experiments, we only exploit encoder-decoder network in our cross-lingual framework.

Finally, we have also incorporated the pre trained general purpose language models such

as ELMo and BERT as the input of the network. Note that to be fully comparable, we freeze the pre trained BERT model so that it can be viewed as an alternative to the static embeddings. We call these embeddings contextualized considering a word can have different corresponding representation vector based on its context. We show the results in Table 4.3.

Task	**Embeddings**		
	ELMo	BERT	Static
WE-fr	9.57	6.47	**15.06**
WE-en	21.39	26.66	**33.47**
BC-en	23.61	26.01	**44.84**
Semeval2013†	**24.279**	3.262	22.003
Semeval2017	**47.703**	29.078	44.382

Table 4.3 – Results of our system with the TF-RNN encoder and wrapped context objective with different embeddings. The pre trained models are presented in Appendix A.5. The WE data set information can be found in Appendix A.1.2 and A.3.1. The Semeval data set information is presented in Appendix A.2.2 and A.3.2.

Recall that the static embeddings for the synonymy task are open domain pre-trained vectors reinforced with specialized domain embeddings, trained on small specialized domain corpora. This solution has been exploited to generate meaningful embedding vectors on specialized domain corpora for bilingual lexicon extraction (Liu et al., 2018; Hazem and Morin, 2017).

The static embeddings concatenated with specialized domain information achieve clearly better results on the specialized domain datasets (WE and BC). On the contrary, the ELMo model holds the best results for general domain corpora. This gives us the intuition that the availability of domain specific information outweighs the choice of a particular word embedding architecture. For a specialized domain corpus, it is more effective to exploit domain information to improve the model rather than using more advanced complicated embeddings.

Results with the BERT model are the worst on the French and the Semeval datasets. Yet, it outperforms the ELMo model on English synonymy datasets. This reveals that the model is less effective on non-English datasets. As for the similarity task, we assume that increasing the training size (e.g., 831 phrases in *Semeval2017* vs 8,923 in *WE-en*) would improve the system because the BERT model use a subword tokenizer that tokenizes often a word into multiple units. This could make it more difficult to generalize meaningful parameter weights during training.

4.6 Synthesis

This chapter completes the the previous chapter on how to train a phrase encoder. We begin with explaining two popular unsupervised training frameworks, the **siamese** network and the **encoder-decoder** architecture.

The second part of this chapter focuses on the **training objectives** for the encoder-decoder architecture as there is a wide variety of suitable training objectives. We present the two most popular objectives: the **autoregression** and the **denoising** approach. To sum up, the autoregression approach predicts the next token according to the previous text tokens, while the denoising approach reconstructs the original input tokens given a corrupted version.

Once we have trained our encoder, we can apply it to our task with two exploitation strategies: the **feature based** and the **fine tuning** approach. Since the two approaches have comparable performance according to the ablation tests presented in Devlin et al. (2018), and the fine tuning approach is much more time and space expensive, we decide to use the feature extraction approach.

Finally, we introduce our proposition, **wrapped context prediction**, which is a new unsupervised training method. The proposed training objective fits well our scenario where we would like to train a phrase encoder with only text information while retaining the syntactic completeness of the original input sentence. Our evaluation experiments have confirmed this intuition. Indeed, results on general and specialized domain exhibit a reasonable performance for the phrase synonymy and similarity tasks.

CHAPTER 5

BILINGUAL WORD ALIGNMENT

In order to align multi-word phrases of different languages, we have to study word-level alignment first. Word-level alignment can be classified into two categories according to the word representation method. As presented in Chapter 2, we will first look at the word alignment using the distributional representation which is followed by the presentation of our contributions to the distributional approach for the bilingual word alignment. Then we address the distributed representation based word alignment. Finally, we present our proposals to improve the distributed representation based approach on both general and specialized domains.

5.1 Distributional representation based approach

Distributional word representations obtained from a word co-occurrence matrix have been applied to word alignment since the 1990s (Fung, 1995; Rapp, 1999). This section introduces word alignment based on the distributional representation whose detail can be found in section 2.1. This approach to word alignment is often called the *standard approach*.

5.1.1 Standard approach

The historical context-based projection approach, also known as the standard approach (SA) has been studied in a variety of works (Fung, 1995; Rapp, 1999; Chiao and Zweigenbaum, 2002; Bouamor et al., 2013; Hazem and Morin, 2016; Jakubina and Langlais, 2017). The first step consists in building the distributional word representation for each language (see Section 2.1 for details).

Next we do the essential step of the standard approach: project the word vector of the source language to the target language space by translating each context via a bilingual lexicon. Let u^w be the word vector for the source word w, and v^w the projected word vector for the source

word w in the target language space. The process can be written as:

$$v_i^w = \sum_{k \in \text{trans}(i)} \frac{1}{D_k} u_k^w \tag{5.1}$$

where i is an index of the distribution word vector, which is actually a target word. k represents a source word which can be translated by i. Since one target word i can be the translation of multiple source words, the value for i in the projected vector v should be the sum of each possible translation u_k. Moreover, each translation's value is normalized by the total number of translations, D_k. Let's see the example in Figure 5.1:

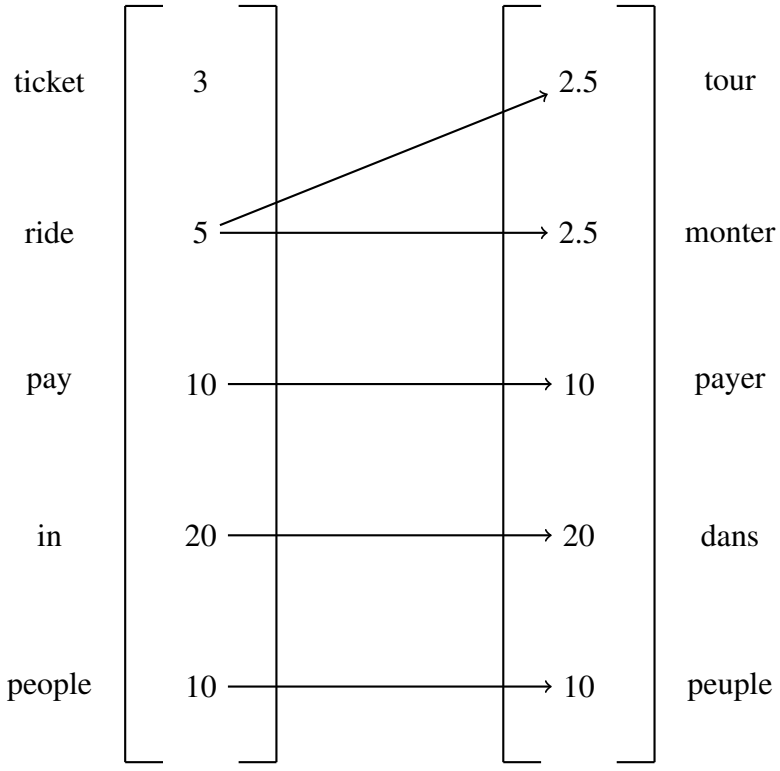


Figure 5.1 – An example of projecting a source word vector to the target language space.

Here in the example, suppose we project an English word vector to the French vector space, and we have a vector $u^{passenger}$ for the source word "passenger". The word "passenger" co-occurs with 5 words ("ticket", "ride", "pay", "in" and "people") within the predefined window in the training corpus. According to the bilingual lexicon we align each non zero value to its corresponding translations. For instance, $v_{peuple}^{passenger} = u_{people}^{passenger} = 10$, and since "ride" has two

translations that are also in the target corpus, we have $v_{tour}^{passenger} = v_{monter}^{passenger} = \frac{1}{2}u_{ride}^{passenger} = 2.5$. It should be pointed out that the translations of "ticket" do not occur in the target corpus, so it will be ignored in the target space vector.

The final step is a comparison process where we compare the projected word vector with all the candidate vectors, which most of the time are the word vectors for all the words in the target corpus vocabulary. Therefore we can rank the candidate words and eventually choose the target word with the highest score as the translation word. In our work, we have chosen the cosine similarity as our comparison score because it is the most widely used score and easy to parallelize because the comparison process with cosine can be calculated by a matrix multiplication between normalized vectors.

To conclude the procedure, we list the steps to implement the standard approach:

1. Construct and normalize the word-context co-occurrence matrix for both the source and target languages. (See Section 2.1 for more details.)

2. Map the source co-occurrence matrix to the target language space via a bilingual lexicon. (See Section 5.1.1 for more details.)

3. Calculate and rank the similarity between mapped source word vectors and target word vectors.

5.1.2 Standard approach with data selection

The standard approach can also be extended by exploiting external data as discussed in Section 2.3, where motivation and detailed steps are explained. The difference between the standard approach and the standard approach with external data lies in the word representation construction. Once the word vectors for the source and the target language are built, we follow the same process as in the standard approach.

In our work, we have chosen the *SSA* (*selective standard approach*) since it has comparable performance with *GSA* (*global standard approach*) and is much more time and space efficient (Hazem and Morin, 2016). In order to validate the effectiveness of SSA, we have evaluated SA and SSA on two corpora, the results are shown in Table 5.1. It is obvious that using external data greatly improves the standard approach for word alignment task.

	SA	SSA
BC	25.9	**56.5**
WE	15.6	**44.4**

Table 5.1 – Comparison (MAP%) between SA and SSA on BC and WE corpus. Details of the corpora can be found in Appendix A.1.1 and A.1.2. The gold standard are the same as in the work of Hazem and Morin (2016) whose details are presented in Appendix A.3.3. We use the *news commentary corpus* presented in Appendix A.2.1 as the external data for SSA.

5.2 Contribution: Standard approach with DSC and WPMI

The key step for word alignment is the word vector representation. We propose two improvements for word co-occurrence vectors. The first addresses the problem of long contexts and the second twist is supposed to fix the over- and under-estimation problem of mutual information.

5.2.1 DSC: distance sensitive co-occurrence

In the standard approach, we note that some context words in the window are not effectively related to the central word. Usually the further the latter is away from a context word, the less they are semantically related. This effect is more obvious especially after stop word filtering. A word originally far from the central word can appear in the context window. This makes the context vector less relevant as a representation for the central word. To reduce this effect, we propose a weighted co-occurrence depending on the distance between the two words, denoted by Distance-Sensitive Co-occurrence (DSC).

$$DSC(w,c) = g(c|w) \times cooc(w,c) \quad \text{where} \quad g(c|w) = \Delta(w,c)^{-\lambda}, \;\; \lambda \in [0,1] \tag{5.2}$$

where w and c respectively denote the central word and the context word, $g(c|w)$ the weight that is distributed to c as the context of w, Δ is the distance between the two words and λ a hyper-parameter that determines the degree of penalization for distant word pairs. Note that λ = 0 is equivalent to a uniform distribution.

5.2.2 WPMI: weighted point-wise mutual information

Another limitation in the standard approach is that MI overestimates low counts and underestimates high counts (Hazem and Morin, 2016). In order to overcome this drawback, we propose the *weighted mutual information* (WMI) inspired by the work of Pennington et al. (2014) where

they introduce a function to smooth word co-occurrences. The original function is a weight function which prevents the overestimation of word co-occurrences. We also use it as a weight function for MI:

$$WMI(w,c) = f(cooc(w,c)) \times MI(w,c)$$
$$f(x) = \begin{cases} (x/x_{max})^{\alpha}, \alpha = 3/4, x_{max} = 20, \text{ if } x < x_{max} \\ 1 \text{ otherwise} \end{cases} \quad (5.3)$$

We have kept the same value for the hyper-parameter α which is $3/4$. Concerning the x_{max}, since our corpora size is much smaller than the one in their work, we decide to make it correspondingly smaller (20). By adding the weight function, the output value for low co-occurrence counts is in fact reduced and the high co-occurrence counts are not impacted because their weight coefficient is always 1.

5.2.3 Evaluation

We have applied our improvements on both the standard approach (SA) and its variant, the selective standard approach (SSA). The experiments are always carried out on the same corpora (BC: breast cancer and WE: wind energy) which have been used for monolingual experiments. The goal is to verify that our proposals improve the word alignment performance with word vector projection based approaches with distributional word representation. Our results are shown in Table 5.2.

We observe in Table 5.2 that WMI alone improves the results compared to those obtained by Hazem and Morin (2016) with MI but also compared to those when using the *Discounted Odds Ratio* (Evert, 2005) as the normalization method, where the MAP is 0.270 for BC and 19.4 for WE (in Figure 2.2). This shows the interest of penalizing small occurrences to compensate the overestimation of the original MI. The second observation is that DSC alone also improves the results. This confirms our intuition that the further a context word is from its central word, the less relevant it is. Finally, we see that combining both enhancements gives the best result. So we could consider that the two enhancements are not mutually exclusive. Another observation is that the improvement of DSC for WE is indeed smaller than for BC, but the tendency is always improving. So we think our hypothesis above holds true for the WE data. Again we can see the same tendency when combining WMI. Note that, despite the difference in improvement between the two datasets, the enhancement that our approach offers is still relevant.

The results in Table 5.2b shows that WMI is less efficient when the data is enriched because

Standard Approach	BC	WE
Hazem and Morin (2016)†	25.9	15.6
SA + WMI	28.9	21.4
SA + DSC	27.4	15.8
SA + WMI + DSC	**29.5**	**21.8**

(a) Result (MAP%) of standard approaches for bilingual word alignment

Selective Standard Approach	BC	WE
Hazem and Morin (2016)†	56.5	44.4
SSA + WMI	55.0	33.9
SSA + DSC	**57.3**	**45.3**
SSA + WMI + DSC	55.8	35.7

(b) Result (MAP%) of selective standard approaches with NC corpus as the external data for bilingual word alignment

Table 5.2 – Results of the word vector projection based approaches with distributional word representation on bilingual word alignment. † indicates results obtained by our implementation of the approach. Details of the corpora can be found in Appendix A.1.1 and A.1.2. The gold standard are the same as in the work of Hazem and Morin (2016) whose details are presented in Appendix A.3.3. We use the *news commentary corpus* (NC) presented in in Appendix A.2.1 as the external data for SSA.

the overestimation of small occurrences is smoothed by the addition of the enlarged overall data as discussed in Hazem and Morin (2016). Moreover, since some words to be translated are quite infrequent or even non-existent in the general corpus, it is possible that penalizing all the small occurrences reduces discriminative features in the general corpus. Besides, although the results of SSA are different the two corpora share again the same tendency: they both reach their best when using DSC without WMI. However DSC always improves the results whether it is applied alone or combined with WMI. As a consequence, it shows that the two enhancements are not mutually exclusive with external data.

5.3 Bilingual word embedding

Word alignment with distributed representations or word embeddings has attracted a lot of attention recently with respect to distributional representations. We name word alignment with word embeddings *bilingual word embedding*. Recall that compared to the distributional word vectors, word embeddings are much lower in dimension. This allows some mathematical operations scalable such as the *singular value decomposition* which is an essential operation for the most popular bilingual word embedding approach. The first part of this section introduces this approach, then we present several improvements.

5.3.1 Language space mapping via a linear transformation matrix

There exists two groups of bilingual word embeddings, the first one relies on some cross-lingual signals such as document-aligned or label-aligned comparable corpora (Søgaard et al., 2015; Vulic and Moens, 2016; Mogadala and Rettinger, 2016), or parallel corpora (Gouws et al., 2015; Luong, Pham, et al., 2015a). We would like to mention that very recently, Lample and Conneau (2019) proposed pre-trained cross-lingual Transformer-based language models using *masked language modeling* like Devlin et al. (2018) and a *translation language modeling* training objective with parallel data to further improve the quality of pre-trained cross-lingual embeddings for languages that share the same alphabet. This is an interesting advancement however the models are not fully released, we will keep an eye on it for our future work.

The second one, however, leverages monolingual corpora to learn separately word representations for each language, and then map the word representations into a common space by a linear transformation with the help of a small bilingual seed lexicon. This approach is pioneered by Mikolov, Quoc V. Le, et al. (2013). A large number of works tried since then to improve the linear transformation method (Lazaridou, Dinu, et al., 2015; Artetxe, Labaka and Agirre, 2016; Liu et al., 2018). Artetxe, Labaka and Agirre (2018a) compiled a substantial amount of similar works (Mikolov, Quoc V. Le, et al., 2013; Faruqui and Dyer, 2014; Xing et al., 2015; Shigeto et al., 2015; Y. Zhang et al., 2016; Artetxe, Labaka and Agirre, 2016; Smith et al., 2017) into a multi-step bilingual word embedding framework. Since the work of Artetxe, Labaka and Agirre (2018a) is more versatile and expressive while reaching the state-of-the-art results, we choose to apply it in our word alignment system. We call it a linear transformation approach.

Data preparation

To implement the linear transformation approach, the first step is to build the word embedding model for both the source and the target language. Then we extract a subset of each model to build two matrices according to a bilingual seed lexicon, X and Z where each row X_i or Z_i corresponds to a word embedding vector. Let v and d be the size of the bilingual lexicon and d the dimension size of the word embedding, $X, Z \in \mathbb{R}^{v*d}$.

Training objective

The original training objective of Mikolov, Quoc V. Le, et al. (2013) is a linear least square problem:

$$\arg\min_{W} \sum_{i} \|X_i W - Z_i\|^2 \tag{5.4}$$

where W is the transformation matrix which allows the product of $X_i W$ to be close to Z_i. Each pair (X_i, Z_i) corresponds to the vectors of a translation pair in the bilingual lexicon. In

the work of Mikolov, Quoc V. Le, et al. (2013) authors use the gradient descent to obtain the transformation matrix. However, by mathematical analysis, we know that the matrix that minimize the problem is:

$$\hat{W} = (X^T X)^{-1} X^T Z \tag{5.5}$$

Later, Xing et al. (2015) introduced an orthogonal mapping to reduce inconsistencies in the previous training objective. Artetxe, Labaka and Agirre (2018a) motivated orthogonality as a way to preserve monolingual invariance, preventing the degradation in monolingual tasks observed for other techniques. And Smith et al. (2017) underlined the fact that the orthogonality is necessary for the mapping to be self-consistent.

By constraining the transformation matrix W to be orthogonal, Equation 5.4 can be written as:

$$\begin{aligned}
&\arg\min_W \sum_i (\|X_i W\|^2 - \|Z_i\|^2 - 2X_i W Z_i^T) \\
&= \arg\max_W \sum_i X_i W Z_i^T \text{ , } \|X_i\| \text{ and } \|Z_i\| \text{ are constant and } \|W\| = I \text{ because of the orthogonality.} \\
&= \arg\max_W \mathbf{Tr}(XWZ^T) \\
&= \arg\max_W \mathbf{Tr}(Z^T XW) \text{ , because } \mathbf{Tr}(AB) = \mathbf{Tr}(BA)
\end{aligned} \tag{5.6}$$

Analytical transformation matrix learning

The optimal solution of Equation 5.6 can be solved analytically if we decompose $Z^T X$ with *singlar value decomposition* (SVD):

$$\text{SVD}(Z^T X) = U\Sigma V^T \tag{5.7}$$

where U, Σ and V^T are intermediate matrices of SVD, and they are all in $\mathbb{R}^{d*d}$. Then $\mathbf{Tr}(Z^T XW)$ in Equation 5.6 can be written as:

$$\mathbf{Tr}(Z^T XW) = \mathbf{Tr}(U\Sigma V^T W) = \mathbf{Tr}(\Sigma V^T WU) \tag{5.8}$$

Since $Z^T X$ is a real matrix, U and V are thus orthogonal matrices by the properties of SVD. Besides, W is also an orthogonal matrix by our constraint. Therefore $V^T WU$ is also orthogonal.

Meanwhile, as Σ is a non-negative real diagonal matrix by the nature of SVD, $\mathbf{Tr}(\Sigma V^T WU)$ will be maximized only if the values in Σ are preserved. This is because each row or column

in the orthogonal matrix $V^T WU$ is a normalized vector by the properties of orthogonality. The product of a non-negative value and a value of a normalized vector is always inferior or equal to its original value. Given what we have discussed, the matrix W that maximizes $\mathbf{Tr}(\Sigma V^T WU)$ is the matrix that makes $V^T WU$ equal to I. Consequently, we have:

$$\begin{aligned} V^T WU &= I \\ VV^T WUU^T &= VIU^T \\ W &= VU^T \end{aligned} \tag{5.9}$$

where V and U can be obtained by applying SVD on $Z^T X$ which is the product of our input word embeddings of each language.

After obtaining the tranformation matrix W, we can project any source word embedding e_s to the target space, obtaining the mapped vector $e_{s->t}$ simply by multiplying it:

$$e_{s->t} = e_s W \tag{5.10}$$

5.3.2 Improvements

There are a substantial body of modifications proposed to improve the linear transformation approach. We summarize several of them.

Length normalization and mean centering

Word embeddings should be normalized to guarantee that each training instance contribute equally to the learning process of the transformation matrix. Xing et al. (2015) introduce the length normalization motivated by the consistency between the Euclidean distance training objective and the vector comparison score which is the cosine similarity. Artetxe, Labaka and Agirre (2016) indicate that when constraining the orthogonality and unit length, maximizing the cosine similarity and minimizing the Euclidean distance are the same training objective. Naturally the length normalization should be applied before the training.

Another modification to the input word embeddings before the training is the mean centering. A centering matrix C_n is a square matrix in $\mathbb{R}^{n*n}$:

$$C_n = I_n - \frac{1}{n} 1_n \tag{5.11}$$

where $\mathbf{1}_n$ is an all one matrix in $\mathbb{R}^{n*n}$. Suppose we have a matrix $X \in \mathbb{R}^{m*n}$, the centering

matrix can be used to subtract the means of rows or columns for X:

$$\begin{aligned} C_m X = X \text{ broadcast minus } \begin{pmatrix} m_{c1}\ m_{c2}\ ...\ m_{cn} \end{pmatrix} &= X - \begin{pmatrix} m_{c1}\ m_{c2}\ ...\ m_{cn} \\ m_{c1}\ m_{c2}\ ...\ m_{cn} \\ ... \\ m_{c1}\ m_{c2}\ ...\ m_{cn} \end{pmatrix} \\ XC_n = X \text{ broadcast minus } \begin{pmatrix} m_{r1} \\ m_{r2} \\ ... \\ m_{rn} \end{pmatrix} &= X - \begin{pmatrix} m_{r1}\ m_{r1}\ ...\ m_{r1} \\ m_{r2}\ m_{r2}\ ...\ m_{r2} \\ ... \\ m_{rn}\ m_{rn}\ ...\ m_{cn} \end{pmatrix} \end{aligned} \tag{5.12}$$

where m_{ci} represents the mean for the i-th column and m_{ri} the mean for the i-th row of X.

For the linear transformation approach, the goal of the mean centering is to subtract the mean of each column of X, because two randomly selected word embeddings (rows) are not similar in any dimension.

Whitening

Matrix whitening is a linear transformation, after which, the covariance matrix of the whitened matrix is supposed to be identity if the original matrix is mean centered, which means that each feature is uncorrelated. Let $X \in \mathbb{R}^{v*d}$ be the matrix that we want to whiten which has already been mean centered by column means, recall that in our case each row represents a word embedding and each column a feature. The covariance of two columns is:

$$\text{COV}(f_p, f_q) = \text{E}(f_p f_q) - \text{E}(f_p)\text{E}(f_q) \tag{5.13}$$

where f_p and f_q are two feature vectors (columns of the matrix X) for any two feature dimension p and q. Since X is mean centered, $\text{E}(f_p)$ and $\text{E}(f_q)$ equals to 0:

$$\text{COV}(f_p, f_q) = \text{E}(f_p f_q) \tag{5.14}$$

Therefore the covariance equals to the correlation. Then we have:

$$\begin{aligned}\mathrm{VAR}(X) = \mathrm{COV}(X, X) &= \begin{pmatrix} \mathrm{COV}(f_1, f_1)\ \mathrm{COV}(f_1, f_2)\ ...\ \mathrm{COV}(f_1, f_d) \\ \mathrm{COV}(f_2, f_1)\ \mathrm{COV}(f_2, f_2)\ ...\ \mathrm{COV}(f_2, f_d) \\ ... \\ \mathrm{COV}(f_d, f_1)\ \mathrm{COV}(f_d, f_2)\ ...\ \mathrm{COV}(f_d, f_d) \end{pmatrix} \\ &= \mathrm{E}(X^T X) - \mathrm{E}(X^T)\mathrm{E}(X) \\ &= X^T X \end{aligned} \tag{5.15}$$

For the whitening, by definition, let M be the whitening matrix for X, it satisfies:

$$\mathrm{VAR}(XM) = (XM)^T XM = I \tag{5.16}$$

This also means that the features are uncorrelated and each feature's variance is 1. Certainly there exists a bunch of matrices that satisfy this condition, we can simply apply the commonly used Mahalanobis or ZCA whitening, where $M = (X^T X)^{-\frac{1}{2}}$. This whitening matrix can be calculated using SVD on X:

$$\begin{aligned} M = (X^T X)^{-\frac{1}{2}} &= ((USV^T)^T USV^T)^{-\frac{1}{2}} \\ &= (V(US)^T USV^T)^{-\frac{1}{2}} \\ &= (VS^T U^T USV^T)^{-\frac{1}{2}} \\ &= (VS^2 V^T)^{-\frac{1}{2}} \\ &= VS^{-1}V^T \end{aligned} \tag{5.17}$$

where U, S and V are the intermediate matrices of SVD on X. Since S is a rectangular diagonal real matrix, S^{-1} is simply obtained by applying the inverse on each value on the diagonal. In addition, it is fairly easy to prove Equation 5.16 with $M = VS^{-1}V^T$:

$$\begin{aligned} (XM)^T XM &= (USV^T VS^{-1}V^T)^T (USV^T VS^{-1}V^T) \\ &= (UV^T)^T UV^T \\ &= VU^T UV^T = I \end{aligned} \tag{5.18}$$

Concerning the bilingual word embedding, we apply the whitening transformation on each

language word embedding matrices **after the length normalisation and mean centering**:

$$\begin{aligned} X_{whitened} &= XM_{ws} \\ Z_{whitened} &= ZM_{wt} \end{aligned} \tag{5.19}$$

where X and Z are word embedding matrices of the source and the target language, M_{ws} and M_{wt} are respectively the whitening matrix for the source and the target language.

Re-weighting

Re-weighting is applied **after the orthogonal mapping**, where we map the two word embedding matrices to a common space:

$$\begin{aligned} X_{mapped} &= X_{whitened}V \\ Z_{mapped} &= Z_{whitened}U \end{aligned} \tag{5.20}$$

where U and V are obtained from $\mathrm{SVD}(Z^T_{whitened}X_{whitened}) = USV^T$. Note that this is a little different from Equation 5.6, where we used one single matrix $W = VU^T$ to apply on Z^TX. In fact, Equation 5.20 represents the same logic because:

$$\begin{aligned} X_{mapped} = X_{whitened}W = X_{whitened}VU^T = Z_{whitened} \\ X_{whitened}V = Z_{whitened}U \end{aligned} \tag{5.21}$$

In addition, when whitening is applied, the cross-covariance is equivalent to the cross-correlation and corresponds to the singular values in S.

The re-weighting transformation distributes some weights to each component where the weights correspond to the cross-correlation (S), increasing the relevance of those that best match across languages (Artetxe, Labaka and Agirre, 2018a). The re-weighting can be applied to the source word embeddings or the target word embeddings.

$$\begin{aligned} X_{reweighted} &= X_{mapped}S \text{ if applied to source word embeddings.} \\ Z_{reweighted} &= Z_{mapped}S \text{ if applied to target word embeddings.} \end{aligned} \tag{5.22}$$

In implementation, we choose to apply the re-weighting to the target word embeddings since it brings better experimental results.

De-whitening

The de-whitening transformation restores the original variances of a word embedding matrix for each feature, this can be done with respect to the original variances in the same language or

to those in the other language since the two matrices are mapped in a common space after the mapping transformation. In practice, we choose to de-whiten the source word embeddings with respect to the source language and vice versa. The de-whitening transformation happens **after re-weighting** and can be implemented as:

$$\begin{aligned} X_{dewhitened} &= X_{reweighted} V^T M_{ws}^{-1} V \\ Z_{dewhitened} &= z_{reweighted} U^T M_{wT}^{-1} U \end{aligned} \tag{5.23}$$

If we put aside the re-weighting which can be seen as a broadcast multiplication with a weight vector, the above equation is equivalent to:

$$\begin{aligned} X M_{ws} V V^T M_{ws}^{-1} V &= XV \\ Z M_{wt} U U^T M_{wt}^{-1} U &= ZU \end{aligned} \tag{5.24}$$

Dimension Reduction

Dimension reduction is applied **after de-whitening**. It keeps the first n features of $X_{dewhitened}$ and $Z_{dewhitened}$. This can be done by multiplying by $0_d + I_n$ which is a $d * d$ matrix with an identity sub matrix in $n * n$. This can also be seen as a special re-weighting because the matrix $0_d + I_n$ is basically a weight matrix where the weights for the first n features are 1 and 0 for the rest.

Evaluation

We have tested several combinations of these modifications on the widely used bilingual word alignment dataset between English and Italian[1]. We report our results in Table 5.3.

whitening	re-weighting	de-whitening	dimension reduction	accuracy
no	no	no	no	39.27
yes	no	yes	no	39.47
yes	yes	yes	no	43.80
yes	yes	yes	150	43.33
yes	yes	yes	200	**44.07**

Table 5.3 – Bilingual alignment results (accuracy) on EN-IT.

We can see that combining all the modifications does bring significant improvements. Note that the word embeddings are pre-trained 300 dimensional vectors, so we tried several different parameter settings for the dimension reduction. We only list the best two settings in the table.

[1] `http://clic.cimec.unitn.it/~georgiana.dinu/down/`

We find that the dimension reduction is only beneficial for the task if it is carefully tuned with the most appropriate parameter. Furthermore, we have different input word vectors in our other experiments which could lead to some inconsistency of the optimal dimension reduction parameter. Meanwhile, the gain is quite marginal if we compare lines 3 and 6 (43.80 vs 44.07). As a consequence, we decide to drop the dimension reduction for our own datasets. Finally, in our system the pipeline is composed of the following steps:

1. Train separately the word embeddings for both the source and target languages. (See Section 2.2.3 for more details.)
2. Length normalisation, mean centering and re-normalisation for the word embedding matrix of each language.
3. Whitening on the word embedding matrices of each language.
4. Orthogonal mapping on the sub-matrices where each row corresponds to a word embedding in the bilingual lexicon.
5. Re-weighting on the word embedding matrices of each language.
6. De-whitening on the word embedding matrices of each language.
7. Calculate and rank the similarity between mapped source word vectors and target word vectors.

5.4 Contribution: Bilingual word embeddings with re-normalisation and data selection

In the original implementation of Artetxe, Labaka and Agirre (2016), we find that after mean centering, word embeddings are no longer in unit length. In order to keep the embeddings length normalized before the mapping, we add a re-normalisation process after the mean centering. Interestingly, in a subsequent work, Artetxe, Labaka and Agirre (2018a) also re-normalize the word embeddings right after mean centering. The effectiveness of the re-normalisation is shown in Table 5.4.

In addition to re-normalisation, following the same idea as in the data selection for the distributional representation, we also would like to exploit external data to enrich our word embeddings. Word embedding systems usually need a large amount of data in order to obtain

Bilingual Word Embedding Approach	Accuracy
Mikolov, Quoc V. Le, et al. (2013) †	34.93
Artetxe, Labaka and Agirre (2016) †	39.27
+ Renorm.	**39.60**

Table 5.4 – Bilingual alignment results (accuracy) on EN-IT with comparison to other approaches. † indicates that the results are obtained from our our implementation of the original work.

reliable word vectors. However the size of domain-specialized comparable corpora is generally very modest (fewer than one million words).

In most of our scenarios, we only have some specialized corpora in small size, which are often not enough for a neural network to generalize meaningful features. The proposal of Hazem and Morin (2016) seeks external general data and learns distributional representations from the sum of all the data. However it occasionally makes the specialized word representation biased by the general corpus. This is especially the case when the specialized corpus contains some infrequent or ambiguous words that have different meanings in different corpora. For instance, the word "farm" in the WE corpus means a site for wind energy production while it is more like to be related to an agricultural site in other domains. Considering these factors, we propose to use a concatenated vector of the one trained on the specialized corpora and on the general corpora as our new word vector. More specifically, we want to concatenate a relatively small size word vector from the specialized corpora and a relatively large size one from the general corpora. In our experiments, the word vector size for the word embedding model trained on the specialized corpus is empirically set to be 100 and the size for the model trained on the general corpus is set to be 300. Hence we preserve the specialized corpus features albeit with less corresponding weight features in the transformation matrix. Consequently the new concatenated word vector carries self-contained information from both corpora (see Figure 5.2). Another advantage is that by doing the concatenation, it is not necessary to retrain our word embedding models over large corpora because we can use existing pre-trained models available. This could save a great deal of time in practice while improving efficiency.

We have evaluated our data selection approach on our own datasets with specialized BC and WE English corpora. We have also tested the re-normalisation on these specialized corpora. The results are shown in Table 5.5.

It is clearly shown that the data selection improve dramatically the performance. In addition to that, we confirm again the effectiveness of the re-normalisation by comparing the last

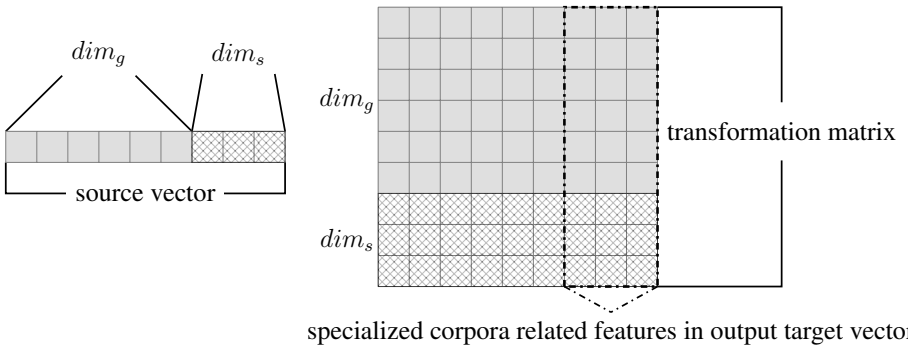


Figure 5.2 – Concatenated word embeddings. Let dim_g be the dimension size of the word vector from general corpora, and dim_s the size from specialized corpora. The transformation matrix will have a total number of $(dim_g+dim_s)^2$ weight features. Among these, $dim_g \times (dim_g+dim_s)$ will be source or target general corpora related features and only $dim_s \times (dim_g + dim_s)$ source or target specialized corpora related ones.

Bilingual Word Embedding Approaches	**BC**	**WE**
BWE	27.4	21.8
Hazem and Morin (2017) ‡	82.3	-
BWE + data selection	82.4	**83.1**
BWE + data selection + Renorm.	**83.2**	**83.1**

Table 5.5 – Result (MAP%) of bilingual word embedding approaches on BC and WE. BWE means our own implementation of bilingual word embedding with the length normalisation and mean centering mentioned in Section 5.3.2. ‡ means that the results are reported from the implementation of the authors, unfortunately it is not publicly available. Details of the corpora can be found in Appendix A.1.1 and A.1.2. The gold standard are the same as in the work of Hazem and Morin (2016) whose details are presented in Appendix A.3.3.

two lines. Our final results in bilingual word alignment on specialized corpora outperform the previous state-of-the-art results in Hazem and Morin (2017). Including the results in Table 5.4 where the corpora is in general domain, we draw the conclusion that re-normalisation improves homogeneously bilingual word embedding with linear transformation approaches. Moreover, combining the re-normalisation and data selection can further improve the performance.

5.5 Synthesis

In this chapter, we reviewed the two major approaches for bilingual word alignment. The approaches of the first class are based on the **distributional representation** while the approaches belonging to the second class exploit the **distributed representation**.

The distributional representation based approach is often called the **standard approach (SA)**. To obtain better results in our experiments, we propose the **distance sensitive co-occurrence (DSC)** during the construction of the word-context co-occurrence matrix to make the word-context co-occurrence vectors sensitive to the distance between a central word and one of its contexts. Then we also propose the **weighted point-wise mutual information (WMI)** in order to rectify the over- and underestimation problem of the original mutual information. Our empirical results prove the effectiveness of these improvements over the SA.

The distributed based approach is also called **bilingual word embedding (BWE)** as word representations are the embeddings. Among different distributed based approaches, we choose to use the linear transformation approach which is introduced by Mikolov, Quoc V. Le, et al. (2013) because it is more flexible and can be more easily extended with out-of-domain data. We also implement several improvements for BWE which are summarized in Artetxe, Labaka and Agirre (2018a).

In line with the **data selection** approach for the distributional approach, we also propose to enrich the word embeddings by exploiting external data. More concretely, we first separately train two word embedding models, next we concatenate word embeddings trained on a corpus in general domain to word embeddings trained on a corpus in specialized domain. The embedding sizes are for instance empirically set. Besides, the **re-normalisation** originally proposed to improve Artetxe, Labaka and Agirre (2016), which is also implemented in Artetxe, Labaka and Agirre (2018a), has proven to be beneficial for the word alignment and it works harmoniously with the data selection.

Our experimental results show that the bilingual word embedding approach surpasses the results obtained by the standard approach. In addition, as for the bilingual word embedding

approach, the data selection significantly improves the performance: +55 points in MAP for the BC corpus and +61 points for the WE corpus. Matrix re-normalization brings an extra minor improvement (on average 0.4 points in MAP). However, it is directly applicable because it does not require any external data.

CHAPTER 6

BILINGUAL PHRASE ALIGNMENT

In previous chapters we have covered the word representations, and word-level alignment with these representations. Besides, we have also introduced the approaches for phrase representation modeling. With all these prerequisites, we can now proceed to address the bilingual phrase alignment (BPA). The simplest way to align phrases is by using a bilingual dictionary, however, this supervised approach eminently depends on the quality of the dictionary. The distributional representation based aligning approach can optimize the dictionary based approach by leveraging the monolingual corpora so that out-of-vocabulary component words can be aligned to a target word and contribute to the phrase alignment. Finally, the unsupervised neural machine translation techniques can be potentially applied to our task so that even without a bilingual dictionary, we can still align phrases of different languages. Besides, with a unified phrase representation, we can then achieve our final objective of aligning phrases of variable length.

6.1 Supervised bilingual phrase alignment with dictionary look-up: compositional approach

The dictionary look-up approach is also called the *compositional approach* (CA) (Grefenstette, 1999; Tanaka, 2002; Robitaille et al., 2006), it is a simple and direct approach that consists in translating each element of a multi-word via a dictionary and generating all possible combinations and permutations. The ranking of the candidates is done by their frequency in the target corpus.

To align a phrase X of length l in the source language to one candidate phrase $\hat{Y}$ in the target language with N candidate phrases, given a dictionary d where one word x can be mapped to a set of k words $\{x_1, x_2, ..., x_k\}$, CA can be implemented by the Algorithm 2. To be clear, $X_{i,k}$ means the k-th possible translation for the i-th component of the phrase X, the total possible translation number k_i depends on both the dictionary entry and the target corpus (all k_i translations must be in the dictionary and the target corpus at the same time).

Algorithm 2: Compositional approach

Data: $X, Y^{(i)}, 1 \leqslant i \leqslant N$
$d : x \longmapsto \{x_1, x_2, ..., x_k\}$, bilingual dictionary
Result: $\hat{Y}$
Initialize a placeholder for the list of possible translations for each component of X, $l_{components}$;
for $i = 1, ..., l$ **do**
  Translate the i-th component of X, X_i by d, obtaining a list of possible translations filtering those not in the target corpus $[X_{i,1}, X_{i,2}, ..., X_{i,k_i}]$;
  Stack the list into the list of possible translations for each component $l_{components}$;
end
Generate all possible combinations from $l_{components}$, obtaining c possible combinations ;
Generate all possible permutations from c, obtaining p;
Rank all the permutations by their frequency in the target corpus;
$\hat{Y}$ equals to the permutation with the highest frequency;

Note that in Algorithm 2, every possible combination c equals to the product of all possible translations:

$$c = \prod_{i}^{l} |\mathrm{trans}(X_i)| \tag{6.1}$$

where $\mathrm{trans}(X_i)$ represents the possible translations for the component X_i.

Moreover, every possible permutation p factorizes again c in terms of the phrase length l:

$$\begin{aligned} p &= c \times \mathrm{P}(l, l) = c \times l! \\ &= \prod_{i}^{l} |\mathrm{trans}(X_i)| \times l! \end{aligned} \tag{6.2}$$

As we can see, the time and space complexity of CA is $O(ll!)$ during the alignment. It is therefore $ll!$ times as complex as a unified representation based approach. And for the permutation generation phase, the time complexity is $O(l^2 l!)$, which is also far more complex than the neural unified representations in Table 3.1. Considering these facts, the compositional approach is not optimized for calculation efficiency.

The compositional approach can only align phrases of the same length. So for instance, "airflow" in English can never be aligned with "flux d'air" in French, which is what we expect in our alignment.

Another remark is that the out-of-vocabulary (OOV) words are not handled by the compositional approach. When a phrase contains at least one OOV word, it is automatically rejected.

Overall, the compositional approach is a simple and quick-to-apply approach with some restraints which prevents us from achieving our final goal in an efficient way. We implement it as a baseline approach.

6.2 (Semi-) supervised bilingual phrase alignment with distributional representation

The main limit of the traditional compositional approach is the inability to translate a term when one of its composing words is not in the dictionary. To solve this problem, Morin and Daille (2012) propose the *compositional approach with context-based projection* (CMCBP), where the objective is to combine the advantages of the standard and compositional approaches by substituting OOV words with their context vectors obtained by the standard approach. This approach allows us to align a phrase having one or even each of its components not in the dictionary. Consequently, the strict need for a fairly complete dictionary is alleviated. We can consider it as a semi-supervised approach with distributional representation.

6.2.1 Compositionnal method with context based projection (CMCBP)

CMCBP begins by building the co-occurrence matrix as in the standard approach. Then it applies a direct translation reinforced by context alignment. If a word of a phrase to be translated is not present in the dictionary, it uses the context vector obtained by the standard approach and projects it into the target language, otherwise it takes the context vector of the target language directly. The next step is the generation of all combinations of possible translation representations for a source language phrase. Finally, the candidate phrases are ranked according to their similarity with phrases of the same length in the target language, and the final score for each possible translation is defined by the arithmetic or geometric mean of each similarity score.

The detailed algorithm of CMCBP is given in Algorithm 3, and an alignment example is illustrated in Figure 6.1. The goal of the example is to calculate the similarity between the French phrase “antécédent familial” as the source phrase and the English phrase “family history” as the target phrase.

In the example, the source component word “antécédent” is not in the given dictionary. We apply SA to obtain its context vector in the target language space V'_{s1}. The other source component word “familial” has two possible translations according to the dictionary, “familial” and “family”. We can extract directly the corresponding context vector V'_{s2_1} and V'_{s2_2} from the

Algorithm 3: CMCBP

Data: $X, Y^{(i)}, 1 \leqslant i \leqslant N$
$d : x \longmapsto \{x_1, x_2, ..., x_k\}$, bilingual dictionary
source and target corpus
Result: $\hat{Y}$
Build the source and target word co-occurrence matrix from the corresponding corpus;
Initialize a placeholder for the list of possible translations for each component of X, $l_{components}$;
for $i = 1, ..., l$ **do**
  if X_i *is in* d **then**
    Translate the i-th component of X, X_i by d, obtaining a list of possible translations filtering those not in the target corpus $[X_{i,1}, X_{i,2}, ..., X_{i,k_i}]$;
    Replace the translated words by their corresponding context vectors $[V_{i,1}, V_{i,2}, ..., V_{i,k_i}]$ by looking up in the word co-occurrence matrix of the target language;
  end
  else
    Apply the standard approach (SA) to obtain the mapped context vector V_i in the target language space, we wrap it into a singleton list $[V_i]$;
  end
  Stack the list into the list of possible translations for each component $l_{components}$;
end
Generate all possible combinations from $l_{components}$, obtaining c possible combinations;
Generate all possible permutations from c, obtaining p;
for $n = 1, ..., N$ **do**
  for $i = 1, ..., p$ **do**
    if $Y^{(n)}$ *has the same length of the* i*-th permutation* **then**
      Initialize a list of score l_score;
      for $j = 1, ..., l$ **do**
        Calculate the score s_j, between the j-th component of the i-th possible permutation and the j-th component of one candidate phrase $Y^{(n)}$;
        Stack s_j into l_{score};
      end
      Calculate the score between permutation i and $Y^{(n)}$ using l_{score}. (arithmetic or geometric mean);
    end
    else
      the score between permutation i and $Y^{(n)}$ is 0;
    end
  end
end
$\hat{Y}$ equals to the candidate phrase with the highest score;

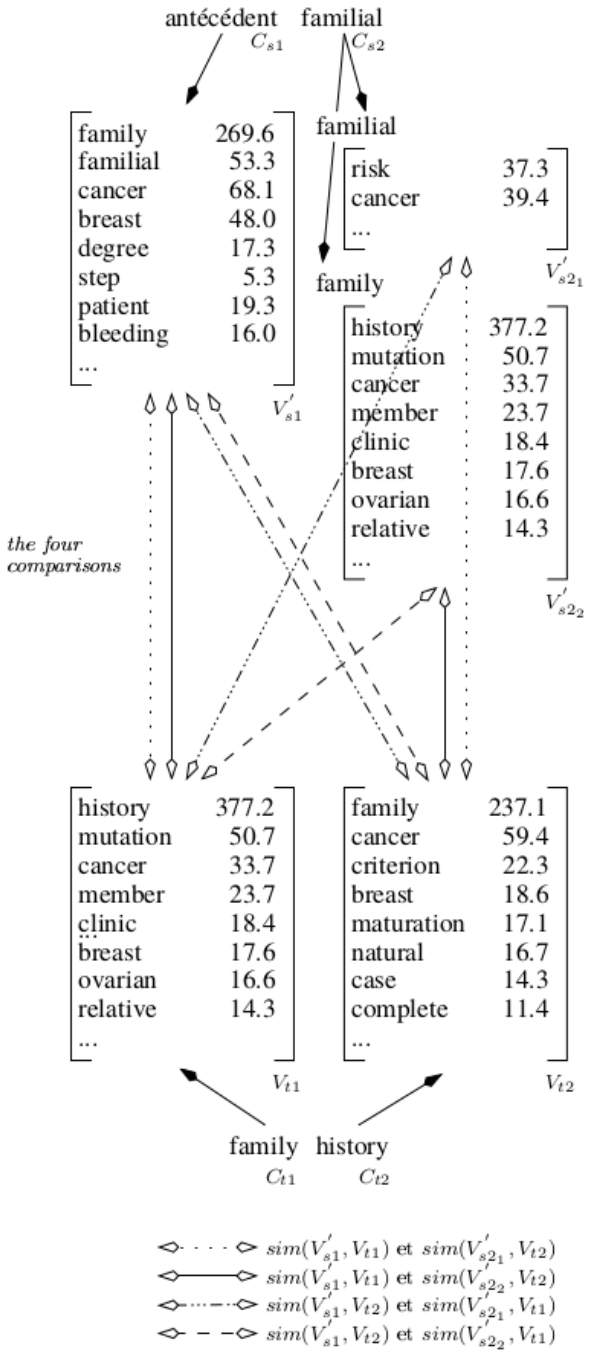


Figure 6.1 – An example diagram of CMCBP. (Source: Morin and Daille (2012))

target word co-occurrence matrix. Finally, we have 4 possible permutations for "antécédent familial" for the similarity calculation.

6.2.2 CMCBP with data selection

The main contribution of CMCBP is the capability of mapping out-of-vocabulary words, however, this requires that the pre-constructed word co-occurrence matrices are complete and reliable. In general, such word co-occurrences are difficult to obtain from a modest size corpus in specialized domain. We can exploit the same data selection strategy as in the selective standard approach (SSA). To do this, we can build the word co-occurrence matrices with external data as in SSA. Then in CMCBP, when a component is not found in the dictionary, we apply SSA with the merged matrices in place of SA. The next steps are the same as in the original CMCBP.

6.3 Contribution : Compositionnal method with word embedding projection (CMWEP)

Following the idea of CMCBP, we propose a to exploit the word embeddings in a compositional approach. The idea is to substitute the word co-occurrence vectors with the word embeddings in CMCBP. We call it *compositional approach with word embedding projection* (CMWEP).

With the word embeddings, we can no longer apply the standard approach to map a source vector to one in the target language space. Nevertheless, we bridge the gap by the linear transformation approach presented in Section 5.3. This requires that we learn the transformation matrix before the phrase alignment.

Besides, one major limit of CMCBP is that it aligns only the phrases of the same length. In order to represent phrases of variable length in a single vector, first we generate all the possible combinations and save their word embeddings. Next we apply the addition to finally retrieve a single vector representing the phrase for one possible combination. Moreover, the addition greatly reduce the complexity of the original CMCBP because we do not have to calculate all the possible permutations. Note that this modification can also be applied to the original CMCBP. We will apply it for our experiments as a good amount of our test samples are pairs of phrases of variable length.

CMWEP can be implemented by Algorithm 4 where we follow the same notation as in Algorithm 3.

Algorithm 4: CMWEP

Data: $X, Y^{(i)}, 1 \leqslant i \leqslant N$
$d : x \longmapsto \{x_1, x_2, ..., x_k\}$, bilingual dictionary
source and target corpus
Result: $\hat{Y}$
Build the source and target word embedding matrix from the corresponding corpus;
Learn the transformation matrix M using the linear transformation approach introduced in Section 5.3;
Initialize a placeholder for the list of possible translations for each component of X, $l_{components}$;
Build the candidate matrix C with addition, where each row is a phrase vector;
for $i = 1, ..., l$ **do**
    if X_i *is in* d **then**
        Translate the i-th component of X, X_i by d, obtaining a list of possible translations filtering those not in the target corpus $[X_{i,1}, X_{i,2}, ..., X_{i,k_i}]$;
        Replace the translated words by their corresponding word embeddings $[E_{i,1}, E_{i,2}, ..., E_{i,k_i}]$ by looking up in the word embedding matrix of the target language;
    end
    else
        Calculate the mapped embedding by BWE, $E_i = E_{s,i}M$, and wrap it into a singleton list $[E_i]$.;
    end
    Stack the list into the list of possible translations for each component $l_{components}$. ;
end
Generate all possible combinations from $l_{components}$, obtaining c possible combinations;
Initialize a placeholder for embeddings of all possible combinations, $A \in \mathbb{R}^{c*d}$;
for $i = 1, ..., c$ **do**
    Calculate the phrase vector by addition;
    Put the phrase vector in the i-th row of A;
end
Calculate the similarity score matrix, S, $S = AC^T$;
$\hat{Y}$ equals to the candidate phrase with the highest value in S where the column index represents candidate phrases;

An example of generating phrase vectors in CMWEP is illustrated in Figure 6.2. Suppose we have the English phrase “onshore wind farm” to align to French. The word “onshore” is not in the given dictionary or the translations are not found in the training corpus. The words “wind” and “farm” have respectively one and three possible translations found. To generate the phrase vectors, we apply BWE on “onshore” and extract the corresponding word embedding for each possible translation of “wind” and “farm”. So each component word vector is mapped to the target language space. By addition, finally we have 3 phrase vectors representing the input phrase.

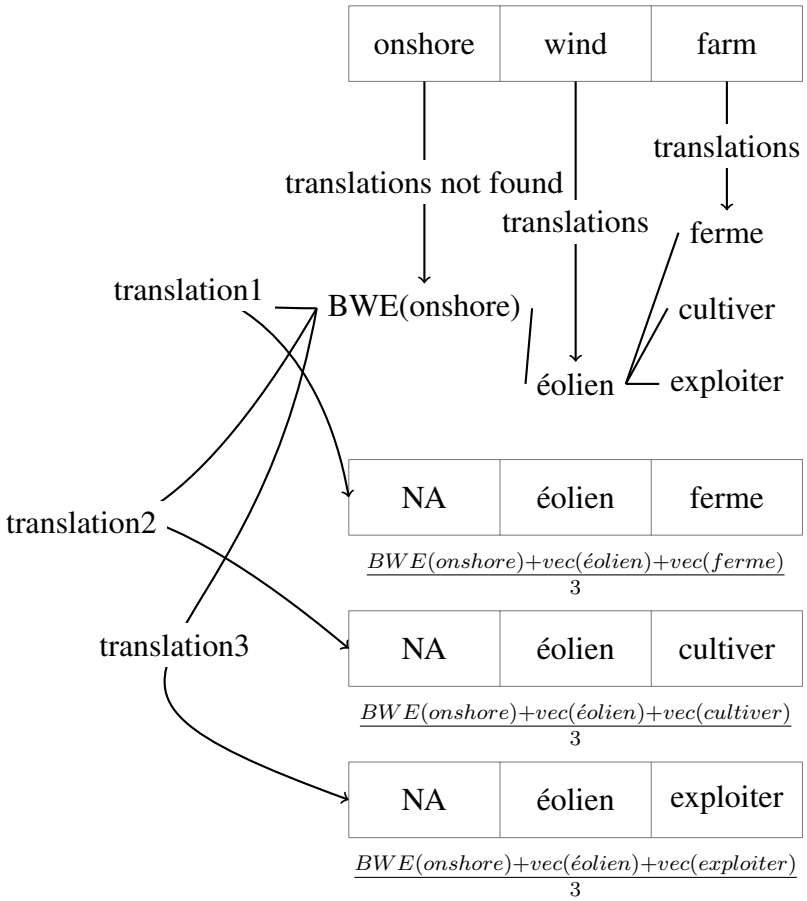


Figure 6.2 – An example diagram of CMCBP.

Evaluation

We include the original compositional approach (CA) and compositional method with context based projection (CMCBP) in our evaluation on the WE corpus. In addition, we also test if the data selection and the techniques presented in Sections 5.2 and 5.3.2 always improve the performance for CMCBP and CMWEP. The results are shown in Table 6.1. Note that we

sum all the possible combinations for CMCBP and CMWEP to obtain one single vector so that phrases of variable length can be aligned.

Model	Accuracy	MAP
CA	59.0	61.5
CMCBP	49.3	61.4
CMCBP + WMI + DSC	46.6	63.2
CMCBP + Data selection	50.7	66.0
CMCBP + Data selection + WMI + DSC	53.4	66.3
CMWEP	52.1	67.8
CMWEP + Data selection	**61.6**	73.3
CMWEP + Data selection + Re-normalisation	**61.6**	**73.4**

Table 6.1 – Accuracy (p@1) and MAP % of different approaches on the WE corpus. Details of the WE-en-fr corpus and its gold standard can be found in Appendix A.1.2 and A.3.3.

First it is to our surprise that the compositional approach (CA) has better accuracy than CMCBP This is because the candidates are ranked by their frequency in the target language corpus while the candidates in CMCBP are ranked by the similarity measure, so theoretically those translated by CA are also translated by CMCBP but with the same similarity score (1.0). Therefore, the final rank in CMCBP is to some degree randomized as multiple top candidates have the same score. However, when using external data, the MAP is considerably improved by CMCBP, this shows that the CMCBP can effectively find many out-of-vocabulary translations that can not be found by CA.

The same problem also exists in CMWEP (randomized rank for the candidates with the same score). Another remarkable point is that unlike the single word bilingual extraction where WMI degrades the results when using external data, combining WMI and DSC improves the results in MAP with or without external data.

Our word embedding methods (CMWEP) notably improve the results when using information carried by external data by almost **12** points in **MAP** and **2.6** points in **accuracy**. Combining the usage of external data and the re-normalization gives the best accuracy and MAP. The characteristics of the WE corpus determine the fact that for most phrases, each component word are included in the dictionary. So CA already provides a high result especially for the accuracy, if we took a less academic or less cleaned corpus for example the *Amazon reviews* where there are more out-of-vocabulary words because of spelling mistakes and internet languages, we expect the result of CA to be much lower while the result of other approaches would

be less impacted. This is also why the re-normalization improves the result very slightly, since most of the translations are found by a dictionary look-up process.

6.4 Unsupervised neural machine translation with pre-trained cross-lingual embeddings

All the previous approaches require a relatively complete dictionary during the alignment, while such dictionaries are not publicly available and sometimes scarce between certain languages. This is especially the case when treating corpora in specialized domain. In order to align phrases with minimal cross-lingual information, we probe a series of unsupervised neural machine translation systems. Although our objective is not a translation task, we can exploit the unsupervised neural machine translation architecture to train our phrase encoder. Then after the training, we can calculate the similarity scores using the phrase vectors generated by the encoder.

6.4.1 Back translation

Neural Machine Translation (NMT) has achieved tremendous progress and recently become the dominant paradigm in machine translation. However, most successful NMT systems are proposed in a supervised or semi supervised context. When no parallel data exists between source and target languages, several works proposed the use of a pivot language (Firat et al., 2016; Saha et al., 2016; Y. Chen et al., 2017) acting as a bridge between source and target. Following the same idea, Johnson et al. (2017) proposed a multilingual neural machine translation model which creates an implicit bridge between language pairs for which no parallel data is used for training. Whether explicitly or implicitly, all these works still require the use of parallel corpora between the pivot language and other languages.

A more recent line of works has completely removed the need of parallel corpora (Lample, Conneau, et al., 2018; Artetxe, Labaka, Agirre and Cho, 2018; Zhen Yang et al., 2018), relying on pre-trained cross-lingual word embeddings presented in Section 5.3 and language models presented in Section 4.3.1 built from monolingual corpora. The loss of the denoising process is:

$$\mathcal{L}_{denoising}(\theta_{enc}, \theta_{dec}) = -\mathbb{E}_{x \in D_l} H(x, dec_{\rightarrow l}(enc(\mathcal{N}(x)))) \tag{6.3}$$

where θ_{enc} and θ_{dec} respectively means the parameters in the encoder and the decoder, $x \in D_l$

is a sampled sequence from the monolingual data and $dec_{\rightarrow l}(enc(\mathcal{N}(x)))$ represents a reconstructed sequence from the noised version of the original sequence x. Apart from the denoising, they all incorporate a *back translation* structure to build pseudo-translations between the source and target language.

Back translation (Sennrich et al., 2016a; J. Zhang and Zong, 2016) has been dominantly used prior to unsupervised NMT systems. It couples the source-to-target translation model with a backward target-to-source model and trains the whole system with a reconstruction loss. In an unsupervised NMT system, the back-translation is activated after a first pass of denoising autoencoding. The gradients for updating the encoder and decoder are calculated by alternatively applying the source-to-target model to source sentences to generate inputs for training the target-to-source model (and vice versa):

$$\begin{aligned} &\mathcal{L}_{backtranslation}(\theta_{enc}, \theta_{dec}) = -\mathbb{E}_{x \in D_{l1}} H(x, dec_{\rightarrow l1}(enc(y))), \\ &\text{with} \quad y = transl(x) = dec_{\rightarrow l2}(enc(x)) \end{aligned} \tag{6.4}$$

where D_{l1} and D_{l2} are the two language corpora, $dec_{\rightarrow l1}$ means that the decoder will decoder the sequence in $l1$ language (or $l2$ resp.). Suppose y is the translation of $x \in D_{l1}$ by applying the decoder of language $l2$, $dec_{\rightarrow l2}(enc(x))$. Then $dec_{\rightarrow l1}(enc(y))$ represents the reconstructed source sentence from the synthetic translation. The goal is to generate pseudo parallel sentence pairs to train the models with a reconstruction loss. Overall, the system is composed of a shared encoder thanks to the pre-trained cross-lingual word embeddings, two specific decoders for the source and the target language and back translation mechanism.

6.4.2 Extract-Edit, an alternative to back translation

Lately, Wu et al. (2019) argue that the popular back translation suffers from the low-quality pseudo language pairs. Since the generated pseudo sentence pairs are usually of low quality, the errors during the back translation training could probably accumulate, leading to a deviation of the learned target language distribution from the real target distribution.

Extract

Rather than using synthetic translations, the authors propose an alternative strategy which extracts potential parallel sentences from comparable monolingual corpora. This is done by simply comparing the sentence encoding generated by the shared encoder. Specifically, for a given source sentence s, the top-k real sentences from the target language space would be extracted based on the $\mathcal{L}_2$ distance. The set of k potential parallel sentences for s is denoted by

M:

$$M = \{t | \min_{1,\dots,k}(||e_s - e_t||), t \in \mathcal{T}\} \tag{6.5}$$

where t is a sentence among all the target language sentences $\mathcal{T}$, e_s and e_t are sentence encodings generated by the shared encoder. In fact, the *extract* can be considered as an alignment process in our definition.

Edit

The objective of the *extract* is to provide some pivot sentences for the system training, however, it is not always guaranteed there the parallel sentence exists in the target corpus. The *edit* operation aims to revise the extracted sentences M in terms of the source sentence s. First, the operation applies a maxpooling layer to reserve the more significant features between the source sentence embedding e_s and the extracted sentence embedding e_t for $t \in \mathcal{T}$. Then the target language decoder will decode it into a new sentence t', forming a new pivot sentence set M':

$$M' = \{t' | t' \in dec(\text{maxpooling}(e_s, e_t)), t \in M\} \tag{6.6}$$

Evaluation network

There is another component which consists in an evaluation network R. After a fully connected layer which takes the source and translated sentence encoding e_s and $e_{\hat{t}}$ as input, it applies a softmax distibution on the similarity between the source s and all the extract-edited target sentences t plus the translated sentence $\hat{t}$ generated by decoding the source sentence encoding, $dec(enc(s))$.

$$P(\hat{t}|s, M') = \frac{\exp\Big(\lambda \cos\Big(s, \hat{t}\Big)\Big)}{\sum\limits_{t^* \in M' \cup \hat{t}} \exp(\lambda \cos((s, t^*)))} \tag{6.7}$$

where λ is an inversed temperature of the softmax function.

Training objective

In addition to the denoising loss, there are several other losses we want to minimize during the training. First the distance of the translated sentence $\hat{t}$ to the source sentence s compared to the top-k extract-edited sentences M' by a comparative loss.

$$\mathcal{L}_{com}(\theta_{enc}|\theta_R) = \mathbb{E}(-\log P(\hat{t}|(s, M'))) \tag{6.8}$$

where θ_{enc} and θ_R are the parameters of the encoder and evaluation network. Combined with

the denoising loss, the overall loss for updating the encoder and decoder is given in Equation 6.9

$$\mathcal{L}_{s \to t}(\theta_{enc}, \theta_{dec} | \theta_R) = w_{denoising} \mathcal{L}_{denoising} + w_{com} \mathcal{L}_{com} \tag{6.9}$$

where $w_{denoising}$ and w_{com} are the hyper-parameters weighing the importance of the denoising and the comparative learning. They are both set to 1 in the work of Wu et al. (2019). Besides, the evaluation network is learned by maximizing the score between the extract-edited sentences M' and the source sentence s. This can be viewed as a "discriminator" in an adversarial training network (Goodfellow et al., 2014), while the decoder-encoder is served as a "generator" which generates a translated sentence $\hat{t}$ with a higher score than the sentences in M'.

$$\mathcal{L}_R(\theta_R | \theta_{enc}, \theta_{dec}) = \mathbb{E}_{t' \in M'}(-\log P(t' | (s, M'))) \tag{6.10}$$

Merging Equations 6.9 and 6.10, the final training objective is:

$$\arg \min_{\theta_{enc}, \theta_{dec}} \arg \max_{\theta_R} \mathcal{L}(\theta_{enc}, \theta_{dec}, \theta_R) = -\mathcal{L}_R(\theta_R | \theta_{enc}, \theta_{dec}) + \mathcal{L}_{s \to t}(\theta_{enc}, \theta_{dec} | \theta_R) \tag{6.11}$$

The whole system is learned by alternatively updating first the evaluation network and then the translation network (encoder-decoder network). Figure 6.3 shows an overview of the whole extract-edit system architecture. Since the work is very recently published and the source code

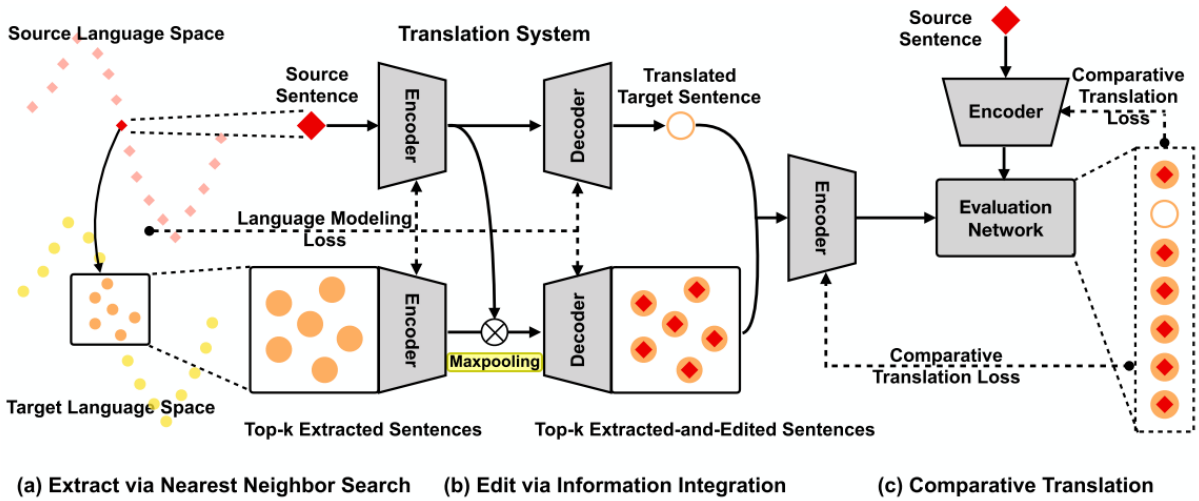


Figure 6.3 – The overview of the extract-edit approach based NMT architecture. (Source: Wu et al. (2019))

is not yet available, we did not include it in our final system. However, the idea of using a comparative model to guide the unsupervised training opens an interesting line of future work.

6.5 Contribution : a new dataset for bilingual phrase alignment

For the bilingual phrase alignment task, we use the same *Wind Energy* (WE) comparable corpus as in the monolingual tasks. This time we evaluate on the English, French, Spanish and Chinese corpora. Besides, we build a new publicly available *Breast Cancer* (BC) English-Spanish comparable corpus by crawling from a scientific website [1]. More specifically, we filter all the publicly accessible articles and then search the keywords "breast cancer" for English articles and "cáncer de mama" for Spanish articles. The final English corpus results from concatenating 168 articles and the Spanish corpus 141 articles. The data crawler script is publicly available [2]. The gold standard is built on the same 108 English phrases as in the monolingual phrase similarity task and the mapping Spanish phrases are manually selected from the target corpus.

The English BC corpus has 26,716 sentences and the Spanish one has 62,804 sentences. For the WE corpus, Hazem and Morin (2016) proposed a reference list consisting of 139 single words for the English-French corpus, while Liu et al. (2018) provided a gold standard with 73 multi-word phrases for the same corpus. Based on the reference list of Liu et al. (2018), we propose a new gold standard including also single words. Moreover, we extended this gold standard to other languages while ensuring that all reference lists share the same 90 English phrases to be aligned. Finally, alignment reference lists were obtained for three languages pairs: English-French, English-Spanish and English-Chinese. For the sake of comparability, we also report results on the datasets of Liu et al. (2018) and Hazem and Morin (2016).

For the preprocessing and phrase extraction, we also use the *IXA pipes* library to tokenize and lemmatize French and Spanish corpora. It is worth noting that the WE Chinese corpus is already pre-segmented. We use the *Stanford CoreNLP* library[3] pos-tagger for all languages, then for the phrase extraction we use the same *PKE* tool as mentioned in monolingual tasks. After hapax filtering, each corpus contains roughly 6,000 phrases of maximal length 7.

[1] `https://www.sciencedirect.com`
[2] `https://bitbucket.org/stevall/data-crawler.git`
[3] `https://stanfordnlp.github.io/`

6.6 Contribution for the unsupervised bilingual phrase alignment

In this section, we present our architecture for unsupervised bilingual phrase alignment which combines the previous techniques. Concretely, the system is an encoder-decoder network with a context prediction training objective. The encoder is a semi tree-free recursive neural network based on the TF-RNN introduced in Section 3.6. We train the network using a pseudo back translation mechanism which grants the ability of being fully unsupervised.

6.6.1 Semi tree-free recursive neural network

To encode phrases of variable length in one single vector, we could have applied the *tree-free recursive neural network* (TF-RNN) introduced in Section 3.6. However, the surprising fact is that in our preliminary experiments, it did not meet our expectation as the synthetic translations are sometimes of low quality and the accumulated translation errors affect more radically with the recursivity (Wu et al., 2019). The same phenomenon also occurs in other similar networks such as the recurrent or LSTM network.

On the other hand, the additive approach (Liu et al., 2018) always successes to hold a decent performance, therefore we decide to adapt the tree-free recursive neural network to the cross-lingual context by levelling the network. Consequently the network has more additive features while being able to distinguish the word order and distribute different weights. More concretely, there are three layers in the adapted version, in the first we operate the same processing as in TF-RNN which consists in splitting the semantics of each word by a linear transformation into two parts: the right side and the left side. Thus the first layer remains a recursive layer. Then we associate these nodes by concatenation, the left side is supposed to be associated with the right side of the previous token and vice versa. The second layer is composed of a fully connected layer that maps the input vectors to output vectors in a specified dimension. Finally the third layer is an addition based pool layer which sums all intermediate level nodes and outputting a single fixed-length vector. The sum operation is motivated by the additive characteristics mentioned in Mikolov, Sutskever, et al. (2013) as the additive approach has showed interesting results in our preliminary experiments. We name it *semi tree-free recursive neural network*. Figure 6.4 shows the schema of the proposed network which is clearly a flat version of the TF-RNN presented in 3.6.

In our scenario we use pre-trained cross-lingual embeddings as the input vector sequence

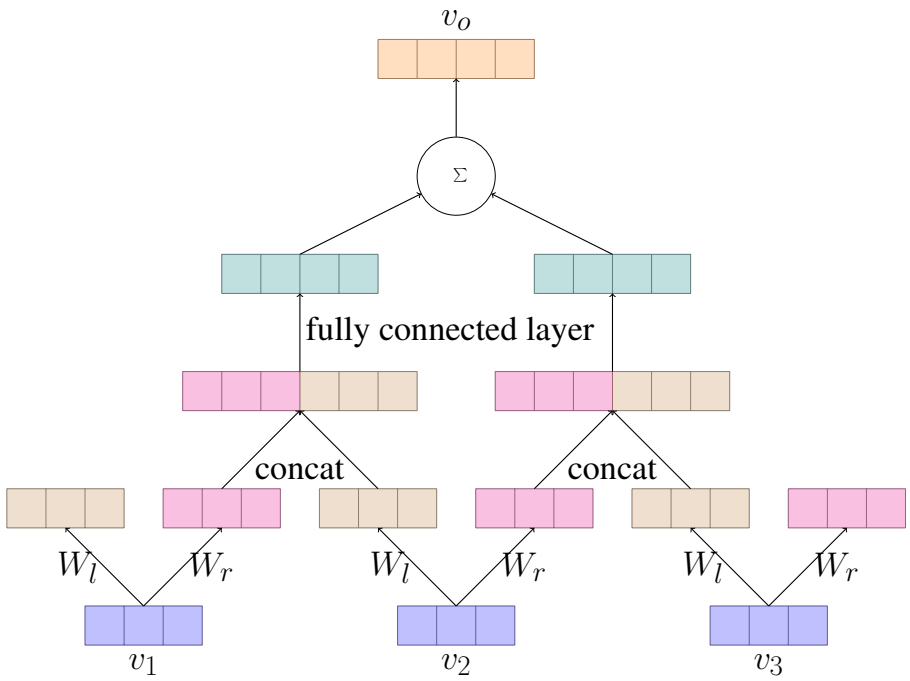


Figure 6.4 – Diagram of the semi tree-free recursive neural network (STF-RNN).

$[v_1, v_2, v_3, ..., v_n]$ with $v_i \in \mathbb{R}^d$, the output vector $v_o \in \mathbb{R}^p$ is calculated as follows:

$$\begin{aligned} v_{i,l} &= \tanh(W_l v_i + b_l) \\ v_{i-1,r} &= \tanh(W_r v_{i-1} + b_r) \\ v_{inter,i} &= \tanh(U[v_{i-1,r}; v_{i,l}] + b) \\ v_o &= \sum_{i=1}^{n} v_{inter,i} \end{aligned} \tag{6.12}$$

where $W_l \in \mathbb{R}^{d*d}$ and $W_r \in \mathbb{R}^{d*d}$ denote respectively the left and the right weight matrices in the linear transformation of the semantic association, $b_l \in \mathbb{R}^d$ and $b_r \in \mathbb{R}^d$ are the corresponding bias vectors, $U \in \mathbb{R}^{p*d}$ and $b \in \mathbb{R}^p$ are the parameters in the fully connected layer with d the input dimension and p the output vector dimension.

6.6.2 Pseudo back translation

Since we do not have cross-lingual data, a direct link between a phrase in language $l1$ and one in language $l2$ is not feasible. However, synthetic translations of single words can be easily obtained using bilingual word embeddings (BWE). By using translated single-word phrases

to train our model, we create stronger links between the two languages. This can be viewed as pseudo *back-translation* as we generate synthetic translations by BWE while in NMT systems the translation is generated by the corresponding decoder (Sennrich et al., 2016a; Artetxe, Labaka, Agirre and Cho, 2018; Lample, Conneau, et al., 2018; Wu et al., 2019). Concerning the training objective, recall the context prediction loss introduced in Section 4.5. Because the encoder never conditions on the context information and the decoder predicts these never seen contexts, we can harmlessly exploit the bidirectional phrase encoders such as PTF-RNN, CNN or Transformer without provoking the redundancy problem mentioned in Devlin et al. (2018). Therefore, the system potentially has four objective loss functions when we alternatively iterate all phrases in the two languages $l1$ and $l2$:

$$\mathcal{L}_{cp\ l1\rightarrow l1}(\theta_{enc}, \theta_{dec_{\rightarrow l1}}) = -\mathbb{E}_{x\in D_{l1}} H(ws(x), dec_{\rightarrow l1}(enc(x))), \tag{6.13}$$

$$\mathcal{L}_{cp\ l2\rightarrow l1}(\theta_{enc}, \theta_{dec_{\rightarrow l1}}) = -\mathbb{E}_{x\in D_{l1}} H(ws(x), dec_{\rightarrow l1}(enc(\text{BWE}(x)))), \tag{6.14}$$

$$\mathcal{L}_{cp\ l2\rightarrow l2}(\theta_{enc}, \theta_{dec_{\rightarrow l2}}) = -\mathbb{E}_{x\in D_{l2}} H(ws(x), dec_{\rightarrow l2}(enc(x))), \tag{6.15}$$

$$\mathcal{L}_{cp\ l1\rightarrow l2}(\theta_{enc}, \theta_{dec_{\rightarrow l2}}) = -\mathbb{E}_{x\in D_{l2}} H(ws(x), dec_{\rightarrow l2}(enc(\text{BWE}(x)))) \tag{6.16}$$

where $\mathcal{L}_{cp\ lp\rightarrow lq}$ means the *context prediction* loss from an encoded phrase in language lp to the context of language lq, $dec_{\rightarrow l}(enc(x))$ is the reconstructed version of the wrapped sentence, $ws(x)$ denotes the real wrapped sentence containing the phrase x and $\text{BWE}(x)$ is the translated single-word phrase for x using bilingual word embedding.

6.6.3 System overview

The general encoder-decoder architecture of our method is shown in Figure 6.5. Since the input sequence is always a phrase, usually much shorter than a sentence, we did not use attention which is intended to capture long-range dependencies. The LSTM decoders predict the context surrounding the input phrase from its encoded vector.

As illustrated in Figure 6.5, in addition to our phrase encoder, we incorporate a pseudo back-translation mechanism for single words based on bilingual word embeddings (Artetxe, Labaka and Agirre, 2018a; Liu et al., 2018). The decoder consists of a single layer LSTM and a fully connected layer on top of it. The goal of the decoder is to reconstruct the *wrapped*

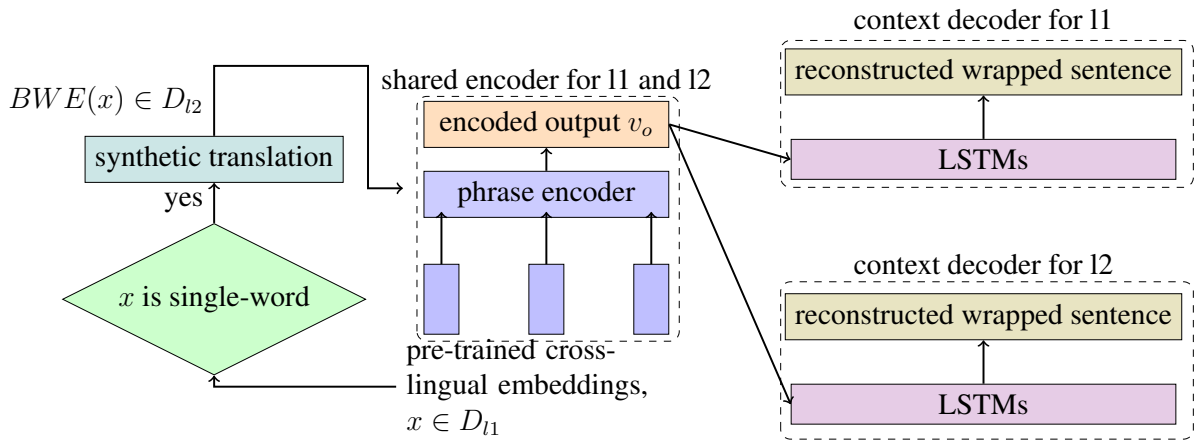


Figure 6.5 – Overview of the cross-lingual alignment training architecture. For a phrase x in language D_{l1}, we first use the shared tree-free phrase encoder, then the system can be trained in two subnetworks: the first one is the encoder-decoder system given the original phrase x w.r.t D_{l1}, and if x is a single-word phrase, we apply a second encoder-decoder system given the translated phrase $BWE(x)$ also w.r.t D_{l1}. We alternatively iterate through all phrases in the two languages. The objective of the decoder is to reconstruct a *wrapped sentence* containing x

sentence which contains the current input phrase. The idea of learning phrase representations based on the context is also studied in Del et al. (2018): instead of an end-to-end system, they first learn all the phrase embeddings by Skip-gram considering them as a single word, and then learn the composition function by a regression model which predicts the pre-trained phrase embeddings from its composing word embeddings. However they limit the phrase length to 2, while we would like to propose a unified end-to-end framework which is able to learn the phrase composition of variable length and the mapping simultaneously.

The detailed training procedure is described in Algorithm 5. Note that in order to keep the training unbiased by the distribution of the number of training phrases in different languages, we build a merged phrase list x with identical phrase number of both the source and the target language. The phrases of the source language are at even indices while the phrases of the target language are at odd indices. We set an early stop condition of three consecutive loss increases.

6.6.4 Evaluation

We evaluate our proposed system on the WE and BC with three different language pairs: English-French, English-Spanish and English-Chinese. The overall results of all test phrases are shown in Table 6.2. We also apply two baseline approaches: the mean vector approach and the distributional word representation based approach, CMCBP. Since the distributional ap-

Algorithm 5: Training process of the proposed unsupervised cross-lingual phrase encoder.

Data: pre-trained transformation matrices for BWE.
source and target corpus.
merged source and target phrase list x.
while *early stop condition not met and max epoch number not reached* **do**
  for $x_i \in x$ **do**
    if x_i *is in* $l1$ **then**
      update the system by Equation 6.13
      $\theta \leftarrow \arg\min_{L_{cpl1 \rightarrow l1}}(\theta)$
      if x_i *is single-word* **then**
        update the system by Equation 6.14
        $\theta \leftarrow \arg\min_{L_{cpl2 \rightarrow l1}}(\theta)$
      end
    end
    else
      update the system by Equation 6.15
      $\theta \leftarrow \arg\min_{L_{cpl2 \rightarrow l2}}(\theta)$
      if x_i *is single-word* **then**
        update the system by Equation 6.16
        $\theta \leftarrow \arg\min_{L_{cpl1 \rightarrow l2}}(\theta)$
      end
    end
  end
end

proach (Morin and Daille, 2012) does not include the alignment of variable length phrases, we ignore the corresponding results in the table.

Dataset				**Method**			
Corpus	Phrases	CMCBP	Mean	RecurrentNN	CNN	LSTM	Proposal
	sw (72)	35.72	47.46	46.71	45.12	46.25	**47.76**
BC	n2n (21)	68.73	81.10	28.52	62.10	50.05	**86.11**
en-es	p2q (15)	-	42.18	1.11	10.65	7.04	**49.11**
	all (108)	-	52.85	36.78	43.04	43.72	**55.40**
	sw (15)	65.56	78.25	77.22	78.33	79.36	**79.44**
WE	n2n (61)	42.09	57.37	6.16	40.84	18.64	**62.19**
en-fr	p2q (14)	-	15.83	<0.5	10.07	9.09	**37.95**
	all (90)	-	55.77	17.25	43.33	27.42	**62.10**
	sw (15)	63.35	77.92	**88.89**	75.78	87.18	87.62
WE	n2n (61)	35.94	**62.68**	7.31	40.33	23.07	61.35
en-es	p2q (14)	-	43.28	<0.5	28.57	17.86	**46.21**
	all (90)	-	62.20	19.77	44.41	32.94	**63.38**
	sw (17)	-	53.43	70.26	**76.47**	71.43	66.50
WE	n2n (47)	-	23.34	17.53	16.55	**25.24**	23.01
en-zh	p2q (26)	-	4.97	5.13	7.60	2.37	**12.32**
	all (90)	-	22.67	23.91	25.28	27.36	**28.13**
WE	n2n (40)	67.32	78.36	46.07	68.51	44.82	**88.01**
en-fr	p2q (33)	-	34.38	2.38	20.01	7.93	**41.83**
Liu et al. (2018)	all (73)	-	58.48	26.06	46.59	28.13	**67.13**

Table 6.2 – Overall MAP % for all phrase alignment. *sw*, *n2n* and *p2q* respectively mean single-word to single-word, same length multi-word and variable length phrase alignment. We do not present the results of the distributional approach on the English-Chinese corpus because we do not have enough resources to build the co-occurrence matrix as in the other language pairs considering the distributional approach requires a high coverage bilingual dictionary, furthermore if the dictionary does not use the same Chinese word segmentation approach as in the WE corpus, it is even harder to find words in it. Details of the WE-en-fr corpus can be found in Appendix A.1.2. For the gold standard, we use the ours presented in Section 6.5 except the last line which applies the gold standard in Appendix A.3.3.

It is clearly shown that the proposed method has a better overall performance. Especially when it comes to different length phrase alignment, the new approach improves significantly the MAP with an average score of 8.8 points. This proves that the proposed method is able to produce high-quality alignment for phrases of variable length. Keep in mind that the differ-

ent length distribution represents a small proportion of all test phrases except for the English-Chinese corpus, so the overall score would be furthermore improved if we have uniform distribution over all kinds of alignment. The second best method on overall results is the addition approach, previously reported to obtain decent results (Mikolov, Sutskever, et al., 2013; Del et al., 2018). Besides, in an unsupervised context, the CMWEP 6.3 is actually equivalent to the addition-based approach. The CNN has some interesting results in same length alignment and the LSTM is powerful at short phrase alignment but unlike in Del et al. (2018), it falls much behind on other types. This difference may be explained by the fact that they limit the alignment to two-word phrases.

The relatively poor results on the English-Chinese corpus may be due to the segmentation of Chinese words. More concretely, as the input vectors for the Chinese sequences are in word-level, many words in our gold standard are not segmented in the same way as in the given corpus which is already pre-segmented. We would like to replace the word-level embeddings by character-level ones or incorporate sub word tokenizer in our future works.

Concerning the single-word alignment on BC, 25 among the 72 single words are in fact acronyms which are particularly difficult to align. This would explain why the single-word alignment has much poorer results than other distributions. Besides, the proposed method obtains strong results for single-word alignment, we believe this happens because the system sees more single-word alignment samples generated by the pseudo back-translation during training.

In order to show that the proposed method can still hold a reasonable performance on single words, we present in Table 6.3 the results for single words compared to state-of-the-art work on bilingual word embedding (Artetxe, Labaka and Agirre, 2018a), including the 139 English-French single word dataset of Hazem and Morin (2016) (suffixed -HM in the Table 6.3). To be comparable, we only test on single-word phrases and the candidate list is limited to all single words in the corpus vocabulary. On our datasets, the source English words are the 15 same as in the *sw* line of Table 6.2.

	BC		**WE**		
Method	**en-es**	**en-fr**	**en-es**	**en-zh**	**en-fr-HM**
(Mikolov, Quoc V. Le, et al., 2013) †	39.96	91.33	87.27	45.88	79.47
(Artetxe, Labaka and Agirre, 2018a) †	**49.13**	**95.56**	**90.39**	**73.52**	**84.01**
Our method	45.96	89.44	88.89	58.75	82.23

Table 6.3 – MAP % for single-word phrase alignment. † indicates that the results are obtained from our our implementation of the original work.

We can see that in general, compared to Artetxe, Labaka and Agirre (2018a), the proposed approach does not degrade much the results except for the English-Chinese words. In addition to that, we succeed to hold a better result with regard to the original transformation matrix method (Mikolov, Quoc V. Le, et al., 2013) with only one exception on the English-French *wind energy* dataset. This shows that our approach is not biased by the compositionality of the multi-word expressions.

For a better understanding of how the proposed method succeeds or fails to align different types of phrases, we analyzed some of the alignments proposed by our system.

Dataset	**Source**	Addition	Our method
BC	breast cancer	cáncer mamario	cáncer de mama
en-es	cell death	muerte celular	muerte
WE	blade tip	angle des pales	côté supérieur de la pale
en-fr	Darrieus rotor	rotor tripale	rotor vertical
WE	airflow	freno aerodinámico	flujo de aire
en-es	wind power plant	electricidad del viento	planta eólica
WE	wind vane	偏航␣电机	风向标
en-zh	electricity power	电力	电力

Table 6.4 – Alignment examples within top 2 candidates ("␣" is the segmentation point for Chinese words)

Table 6.4 shows examples extracted from top 2 nearest candidates to the source phrase in column 2. Again we see that the proposed method is capable of generating better results over different types of alignment. In the first example, with the proposed approach, the source phrase *breast cancer* is aligned to *cáncer de mama* (lit. "cancer of breast") which is the expected phrase in Spanish and is far more idiomatic than *cáncer mamario* (lit. "cancer mammary") obtained by the addition approach. In line 7 we see that the perfect translation for *wind vane* is found by our proposal: 风向标, while the additive approach finds 偏航␣电机 (lit. "yaw electric machine"). Besides, examples in lines 3, 5, 6, 7 and 8 are all composed of phrases of variable length, the corresponding reference phrase can be found in the fourth column. Interestingly, we find that the proposed system find paraphrases referring to fairly domain-specific phrases like *blade tip* which is aligned to *côté supérieur de la pale* (lit. "side top of the blade"). This is also the case for *Darrieus rotor* aligned to *rotor vertical*, which is remarkable since the Darrieus rotor is a kind of vertical rotor.

Though the proposed method performs generally well on phrases, we observe that it em-

phasizes occasionally too much the syntactic head in a multi-word phrase. For instance, in the second example, *cell death* is aligned to *muerte* ("death"), while the addition based approach succeeds to align it to *muerte celular* (lit. "death cellular") which is the reference phrase in Spanish. Undoubtedly, *death* is the syntactic head for the noun phrase *cell death*, it is clear that the proposed method puts more weight on the syntactic information rather than the compositional property for this phrase. This also explains why we do not obtain better results on equal-length phrase alignment on the English-Spanish and English-Chinese *wind energy* corpora (Table 6.2). This bias could be due to the increased amount of single-word phrase samples of the pseudo back-translation reinforced learning. This suggests that we could possibly improve the system by adding synthetic translations for multi-word phrases during the training.

6.7 Synthesis

Unsupervised bilingual alignment of variable length phrase, which is also our main goal, is finally addressed in this chapter. Following the same line of the supervised baseline approach, **compositional approach (CA)**, Morin and Daille (2012) propose to remove the constraint of aligning only in-vocabulary words by exploiting the distributional word representation. The approach is an extended version of CA hence it is called **compositional method with context based projection (CMCBP)**. With the distributional word representations, CMCBP turns into a semi-supervised approach. Although it is a step forward towards unsupervised phrase alignment, it is not when it comes to the unified phrase alignment because it can only align phrases of same length.

To align phrases of variable length, we can simply sum all the component word vectors of a phrase. As discussed in Section 3.1, the addition based approach is a simple yet effective way to encode multiple word vectors into one single vector, allowing comparisons between phrases of variable length without introducing any further parameters. Therefore, we apply the addition approach in CMCBP to make it compatible with all phrases. In addition, we also propose to substitute the distributional word vectors with word embeddings, which is named **compositional method with word embedding projection**. In our experiments we find that incorporating word embeddings improves the phrase alignment performance by about 7 points in MAP.

It's worth mentioning CMWEP in an unsupervised context is identical to a simple similarity comparison task with addition encoding. In Section 3.3 we point out that the addition approach ignores the inner structure of multi-words, and the proposed TF-RNN (see Section 3.6) can

encode phrases according to a certain learned structure without introducing any supervised information such as the parsing tree. Therefore, we would like to improve the performance by exploiting the TF-RNN with the context prediction training objective (see Section 4.5). Now the only barrier which keeps us from reaching our final goal lies in the parallel data. Thanks to the unsupervised machine translation technique, **back translation** (Sennrich et al., 2016a; J. Zhang and Zong, 2016), we can create some synthetic parallel text during the training process. Combining all these, we build an encoder-decoder network with TF-RNN as the phrase encoder, an LSTM layer as the decoder and a back-translation-like mechanism where the synthetic translation is generated by the bilingual word embedding instead of the decoder in an NMT system. The preliminary experiments have shown very poor results because the training is biased by the single words and the accumulated translation errors affect more radically with the recursivity (Wu et al., 2019). We introduce a flatten version of TF-RNN to replace the fully recursive version because we notice that the additive approach always successes to hold a decent performance. The idea is to reduce the recursivity of the phrase encoder so the errors produced by BWE are not propagated to every node.

Finally, our system outperforms the addition approach across all our datasets. And our proposed phrase encoder obtains better results compared to other state-of-the-art architectures such as CNN and LSTM. The gain is especially significant for the different length phrase alignment with improvements from about +10 to +40 in MAP.

CHAPTER 7

CONCLUSION AND PERSPECTIVE

In this final chapter, we first give an overview of what has been studied and proposed. Then we open up perspectives for the future work based on the observation of our experiments.

7.1 Conclusion

Significant advances have been achieved in bilingual word-level alignment from comparable corpora, yet the challenge remains for phrase-level alignment. Traditional methods of phrase alignment can only handle phrases of equal length, while word embedding based approaches learn phrase embeddings as individual vocabulary entries suffer from a data sparsity problem and cannot handle out of vocabulary phrases. Our objective in this thesis is the unsupervised bilingual phrase alignment which is a comparative task where a unified phrase representation plays a key role.

In order to learn unified phrase representations for variable length phrases, we have studied the approaches for unified phrase modeling and cross-lingual phrase alignment, ranging from co-occurrence based models to most recent neural state-of-the-art approaches. Besides, since our training corpora in specialized domain are modestly sized, we exploit external general domain data in our framework. After our survey on existing approaches, we propose a new architecture called *tree-free recursive neural network* (TF-RNN) for modeling phrases of variable length which, combined with a wrapped context prediction training objective, outperforms the recent state-of-the-art approaches on monolingual phrase synonymy task with only plain text training information. We attribute the high performance to several points:

- **Compositional**. Like the addition approach, TF-RNN is also a compositional approach where component token vectors are associated to form the final phrase vector. Furthermore, by recursively applying weight matrices for each combination pair, the final phrase vector takes word inner relation into account.

- **Sequential**. Like RecurrentNN, TF-RNN is also a sequential approach where unlike the

addition approach, component order counts. However, it is best suited to short sequences such as phrases. Instead of depending more on the latest parsed words in RecurrentNN, the final output vector of TF-RNN depends more on the higher level representations of the phrase, which we think coherent because humans process natural language following a syntactical tree and the latest words they see are not always those who have the largest weights.

- **Tree-free**. Like RNN, TF-RNN also generates the phrase vector following a tree structure, however, it removes the need of an input tree structure by systematically building a recursive binary tree.

- **Context-aware**. Like the extended word embedding approach, we train our TF-RNN by leveraging the phrase context in order to learn from the information outside the components. While the extended word embedding approach considers phrases as one single unit and suffers from the data sparsity and out-of-vocabulary problems, TF-RNN can still be applied to new freely combined phrases thanks to its compositional properties.

As for the Transformer based encoder, its low performance in our phrase synonymy experiments are indeed surprising to us. We assume that the multi-head attention is probably more suitable for long sequence tasks as the advantage of the attention based models lies in the caption of long range dependencies.

For the alignment part, we have reviewed supervised and unsupervised training frameworks with the traditional compositional approaches and the recent neural machine translation systems. Finally we propose to incorporate an architecture called *Tree-free semi recursive neural network* (TF-SRNN) derived from TF-RNN in an *encoder-decoder* model with a pseudo back translation mechanism inspired by works on unsupervised neural machine translation. The system can be trained without cross-lingual phrase table and different length phrases are encoded and represented in a homogeneous way. Empirically, our proposition improves significantly bilingual alignment for phrases of different lengths while achieving state-of-the-art results on single-word and same length multi-word phrase alignments.

7.2 Operational contributions

During this thesis, our implementations has been partially industrialized as a component of the company's software. In the meantime, we have tightly followed the development and road map

of the framework *deeplearning4j*. We have made a contribution to this project which added a new neural network class.

7.2.1 Industrialized implementation

A first part of our work has already been integrated in the product which involves the topic modeling task. More specifically, we exploit the addition based phrase representation of component word embeddings (Section 3.1.2) along with the concatenation based data selection (Section 2.4). Then, *principal component analysis* (PCA) can be applied to retrieve the most discriminant features of the mean centered phrase vectors. Finally, the phrases are unsupervisedly classified into several topics using k-means clustering. These functions are implemented with the JVM mathematical library *Nd4j*.

7.2.2 Contribution to the JVM based deep learning framework

We have paid close attention to the roadmap of *deeplearning4j* which was a first choice for JVM deep learning. However, when we began with this framework with the previous versions ($<= 0.91$), it did not support the popular auto-differentiation so each time we want to make some custom neural network layer we have to implement it manually. The implementation includes the initialization of the parameters, the forward pass and the backward pass where the gradients are manually calculated. We have implemented an element-wise multiplication layer and contributed it to the project.[1]

7.3 Future work

This work opens exciting opportunities for future research, as we mentioned through the manuscript that there are still several aspects with substantial room for improvement. Particularly, we identify the following future works.

7.3.1 Data selection refinement

We exploit external data to reinforce our word-level representations due to the word co-occurrence inconsistency caused by the modest size of the training corpora. We show that the best result is obtained by the concatenated word embeddings. In our experiments, we empirically set the

[1] `https://github.com/eclipse/deeplearning4j/pull/4255`

dimension for the word embeddings trained from the general and specialized domain corpora. Although it is true that the training of a neural network can adjust the weights associated to each word embedding dimension, we still believe that a smarter way of leveraging the external data would bring further improvement. For instance, some polysemy words have a different meaning in a specialized domain corpus compared to its most common meaning in a general domain corpus. If we set the dimension of the general domain embedding higher than the specialized domain embedding, then we probably will only capture its most common interpretation which does not correspond to its real meaning for tasks in specialized domain. Otherwise, if the specialized domain embedding is too important, it will be even worse since most words keep their most common interpretation in a specialized domain corpus and the embeddings trained on large size general domain corpus are more reliable and coherent.

Two strategies could be more deeply explored for future work: pre-training corpus merge and post-training embedding merge. Note that the latter is actually applied in this thesis. The first strategy investigates the nature and the quality of the corpora and trains the word embeddings with one finely merged corpus. The second one trains separately word embeddings from the general and specialized domain corpora, and then merges these word embeddings with some operations such as the concatenation incorporated in this thesis. Interestingly, Schuster et al. (2019) introduced *embeddings anchor* which exploited contextualized word embeddings such as the pre-trained *ELMo*. The goal of the *embedding anchor* is to generate a context-independent word embedding from the contextualized models so that the richer information can be exploited for word alignment and polysemy words can be linked to its different meanings. We would like to study the behavior of using different merging approaches such as the approach of Schuster et al. (2019), a specific layer related to the merge or multi-task learning for both separated embeddings.

7.3.2 Synthetic multi-word generation

As a means to train the phrase alignment system, we choose to incorporate back translation in our framework. The idea is to generate synthetic translations for single-word phrases since the pre-trained bilingual word-level embedding model produces more accurate word translations. We could have the decoder generate synthetic translations for multi-word phrases but it would largely degrade the performance because of the size and the comparability of our training corpora. Nevertheless, the translation errors produced by the bilingual word embedding process are propagated and hence accumulated through the training samples. Consequently the system performance is still somewhat distorted by these errors.

Concerning future works, we would like to further study a recent alternative approach to back translation, extract-edit (Wu et al., 2019). Based on a similar idea, we could use extracted and potentially edited phrases as the synthetic translations which would avoid the misleading caused by the poor generated synthetic translations and fill the blank of synthetic multi-word translations.

7.3.3 Contextualized cross-lingual input

With the release of the deep contextualized pre-trained cross-lingual language model(Lample and Conneau, 2019), apart from the pre-trained static cross-lingual embedding, we can also apply the contextualized cross-lingual embedding as our input. Furthermore, since the contextualized embedding are basically sequential model, we can use it as a shared encoder and fine tune the whole network. However, this means that all the parameters in the language model need to be calculated during the backward pass. The training process would take much more space and time than the feature extraction scenario plus a small network such as the TF-RNN.

7.3.4 Applications beyond phrase alignment

The output of our work can be viewed as a phrase mapping table. Since such resource is crucial to SMT systems, we hope to leverage our output in an SMT system or other down stream tasks beyond the phrase synonymy and similarity tasks, with or even without adaptation. Comparing the performance between our phrase table based SMT system and an unsupervised NMT system would yield more insight on unsupervised or low-resource training studies.

7.3.5 Model deployment in industrial context

Since our objective is to provide reliable mono- and cross-lingual phrase representations for real world scenarios, it is once the model is deployed to production that our objective is fully achieved.

We mentioned above that a part of our work has been incorporated in the product. However, because our models are initially designed for an experimental usage, two essential functions are lacking to obtain an ideal machine learning solution:

- On-the-fly inference optimization. For most cases, the model in production should be optimized for an on-the-fly inference, which includes the interaction with the database, user request design, etc.

- Continuous self-learning pipeline with client feedback. User feedback is a precious resource for improving the machine learning model as it builds a task-specific supervised data set without too much extra cost. A pipeline allowing to leverage the feedback is beneficial to both the service provider and the user.

This line of future work involves a thorough design of the machine learning architecture and co-working with others. We hope to wrap our models in a continuous self-learning machine learning pipeline in the future.

APPENDIX A

RESOURCE AND DATA

A.1 Corpora in specialized domain

In this section, we present specialized domain corpora which existed prior to this thesis, and that were used by several past studies.

A.1.1 Breast cancer (BC)

This corpus is composed of documents collected from the Elsevier website[1]. The documents were taken from the medical domain within the subdomain of breast cancer. There are English and French comparable corpora where the English corpus has 525,934 tokens and the French 521,262 tokens.

A.1.2 Wind energy (WE)

This corpus has been released by the TTC project. The corpus has been crawled using the Babouk crawler (Groc, 2011) based on several keywords such as "*wind*", "*energy*", "*rotor*" in English and their translations in French.[2] The whole data set has 6 languages, we only use the English, French, Spanish and Chinese corpora. They respectively have 313,943, 314,549, 453,953 and 487,286 tokens.

[1] http://www.elsevier.com
[2] http://www.lina.univ-nantes.fr/?Reference-Term-Lists-of-TTC.html

A.2 Corpora in general domain

A.2.1 New commentary (NC)

This corpus consists of political and economic commentaries crawled from the web[3]. We use this corpus as the external data for the distributional approaches in English and French. The English and French corpora have respectively 5.7M and 4.7M tokens.

A.2.2 Semeval 2017

This English corpus is only used for Semeval 2017 task2[4], it is extracted from *Wikipedia* and has 1.7 B tokens. Note in practice we only use a subset of the corpus where each sentence contains at least one phrase in the test list.

A.3 Gold standard and candidate list

A.3.1 Phrase synonymy

A list of phrases is extracted using the open source tool PKE[5] for the the three phrase synonymy datasets: WE-English, WE-French and BC-English. Each corresponding candidate list has 8,923, 6,412 and 8,989 phrases. PKE is a symbolic terminology extraction system which generally extracts three times as many multi-words as single-words in the vocabulary. The terms are extracted following some pre-defined syntactic patterns, for example the pattern *NOUN NOUN* could lead to the extraction of *shop assistant*. As for WE-English and WE-French, we use the same gold standard as in Hazem and Daille (2018) with respectively 49 and 14 phrases. The small gold standard size problem is also mentioned in the original work. For the BC corpus, we have presented it in Section 3.5.

A.3.2 Phrase similarity

The phrase similarity gold standard is directly obtained from the official website of the Semeval tasks[6]. Since most phrase pairs of the gold standard of Semeval 2017 task2 are single-word

[3] http://opus.lingfil.uu.se
[4] alt.qcri.org/semeval2017/task2/index.php?id=data-and-tools
[5] github.com/boudinfl/pke
[6] alt.qcri.org/semeval

pairs, we have filtered the list to obtain a gold standard where each phrase pair contains at least one multi-word phrase. Finally there are 95 of such phrase pairs. The Semeval 2013 task5 gold standard was originally a binary reference where 7,814 phrase pairs are tagged with "positive" or "negative" for the similarity. We adapted it to a similarity list with 1 for positive and 0 for negative.

A.3.3 Phrase alignment

Apart from the new reference list that we have built, the reference lists for single-word phrase alignment for BC and WE corpora are the same as used in Hazem and Morin (2016) which consist of 248 word pairs for BC and 139 for WE. The reference list for unified multi-word phrase in WE is built based on the term list provided by the project site. Finally this list contains 73 phrase pairs but each pair has multiple variant translations and in our settings, we consider them to be also the gold translations[7]. The list is built based on 277 one-to-one mapping pairs, but if we use directly this list the results would be biased by those who have multiple translations, therefore we factorized these pairs and finally obtained a list of 73 one-to-many mapping pairs. The reference list for the Italian/English task is the same as in Artetxe, Labaka and Agirre (2016) which contains 1,500 entries.

The candidate list is also extracted from PKE following the same pipeline as in the monolingual tasks. Therefore the WE-English and WE-French corpus has always 8,923 and 6,412 phrases as for the phrase synonymy task. The WE-Spanish and WE-Chinese corpus have respectively 8,323 and 5,747 phrases.

A.4 Bilingual dictionaries

For our French/English experiments, we use the French/English dictionary ELRA-M00337[8] (243,539 entries). From this dictionary we select a subset of 3,007 entries from the BC corpora and a subset of 2,745 entries from the WE corpora based on a word frequency threshold of 5. These two subsets are used as the training data in our word embedding mapping experiments. For our Italian/English experiments, we only use the same seed lexicon[9] as used in Artetxe, Labaka and Agirre (2016) where 5,000 entries are manually selected.

[7]The reference list and evaluation software are available here: `https://github.com/Dictanova/term-eval`

[8]`http://catalog.elra.info/product_info.php?products_id=666`

[9]`https://github.com/artetxem/vecmap`

A.5 Pre-trained models

A.5.1 Word embeddings

We have used pre trained word embeddings in English, French, Spanish and Chinese. They are all 300-dimensional *Fasttext* vectors[10] pre trained on wiki dump. Regarding the embedding models of the domain specialized corpora, we use *deeplearning4j*[11] to train domain-specific 100-dimensional word embeddings using the Skip-gram model, with 15 negative samples and a window size of 5.

A.5.2 Language models

We have incorporated the implementation of BERT[12] and ELMo[13] because they both have pre-trained models on multiple languages. The BERT implementation has a multilingual model which contains 104 languages while the ELMo implementation has 44 separate language models. All these models are pre-trained on wikidump plus common crawl corpora (1B words for ELMo and 3.3B for BERT).

[10] http://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

[11] deeplearning4j.org

[12] github.com/huggingface/pytorch-pretrained-BERT

[13] github.com/HIT-SCIR/ELMoForManyLangs

APPENDIX B

EXPERIMENT SETTINGS

B.1 Implementation tools

Through our experiments, we have used two major frameworks.

- *Deeplearning4j-beta*[1]. The *deeplearning4j* framework is a JVM based machine learning tool which has implemented some of the most popular models such as *word2vec* and provides a numpy-like tensor library *Nd4j* in native JVM environment. In our early experiments, we used this framework to learn monolingual and cross-lingual word embeddings and build the distributional representation based approaches.

- *Pytorch-1.2*[2]. Pytorch is a dynamic computation graph based neural machine learning framework in Python. Unlike *tensorflow 1.0* or *deeplearning4j* where we define computation graphs statically before a model can run (discussed at the end of Section 2.2.1), Pytorch uses dynamic computational graphs where each run of the graph can be different. More specifically, for some models we may wish to perform different computations for each data point. For a dynamic computational graph this can be done using imperative flow control while in a static graph we have to declare different paths at the graph build step.

B.2 Distributional representation setting

In the standard approach, the window size is 3 (a total of 7 words are considered), which is the same as in Hazem and Morin (2016). The distance weight parameter λ in distance-sensitive co-occurrence is empirically set to be 0.25. For the *weighted mutual information*, we have kept the same value for the hyper-parameter $\alpha = \frac{3}{4}$ as in the work of Pennington et al. (2014).

[1] `https://deeplearning4j.org/`
[2] `https://pytorch.org/`

Concerning x_{max}, since our corpora size is much smaller than the one in the original work, we decide to make it correspondingly smaller (20).

B.3 Encoder-decoder network setting

Our models are trained for a maximum of 200 epochs with an early-stop condition of three consecutive loss increases. We use the *Adam algorithm* with *AMSGrad*[3] to update the models. The initial learning rate is $1e-4$. One model with static word embeddings takes roughly 2 days to train on a single Geforce 1080 Ti GPU with Pytorch 1.0 and Cuda 10 on Ubuntu 16.04, while training with contextualized embeddings (ELMo/BERT) takes about 4 days with the feature based strategy.

We pad the special token `[CLS]` to the beginning of every sentence for the models with BERT. To extract the features from the BERT model, we sum the output vector of the last four hidden layers (Devlin et al., 2018), this has shown to be the second best method, with only 0.2 F-score point behind concatenating the last four layers which is 4 times less space efficient. The generated phrase vectors are compared by cosine similarity. For the synonymy task we simply calculate the cosine of all pre-extracted phrase candidates. We use the evaluation script provided with the *Semeval2017* dataset for the similarity task, and the MAP score (Christoper D. Manning et al., 2008) to evaluate the synonymy task:

$$MAP = \frac{1}{|W|}\sum_{1}^{|W|}\frac{1}{Rank_i} \tag{B.1}$$

where $|W|$ corresponds to the size of the evaluation list, and $Rank_i$ corresponds to the ranking of a correct synonym candidate i. Besides it is also worth noting that the baseline for this task (Camacho-Collados et al., 2016) uses additional data like a concept net and a different pre-trained word embedding model, so it is not comparable with systems which only use provided text data.

[3] `https://openreview.net/forum?id=ryQu7f-RZ`

BIBLIOGRAPHY

Artetxe, Mikel, Gorka Labaka and Eneko Agirre (2016), "Learning principled bilingual mappings of word embeddings while preserving monolingual invariance", *in*: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*, Austin, TX, USA, pp. 2289–2294, URL: https://aclweb.org/anthology/D16-1250.

— (2018a), "Generalizing and Improving Bilingual Word Embedding Mappings with a Multi-Step Framework of Linear Transformations", *in*: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, New Orleans, LA, USA, pp. 5012–5019.

— (2018b), "Unsupervised Statistical Machine Translation", *in*: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP'18)*, Brussels, Belgium, pp. 3632–3642.

Artetxe, Mikel, Gorka Labaka, Eneko Agirre and Kyunghyun Cho (2018), "Unsupervised Neural Machine Translation", *in*: *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*, Vancouver, Canada.

Axelrod, Amittai, Xiaodong He and Jianfeng Gao (2011), "Domain Adaptation via Pseudo In-Domain Data Selection", *in*: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP'11)*, Edinburgh, Scotland, UK., pp. 355–362, URL: https://www.aclweb.org/anthology/D11-1033.

Bahdanau, Dzmitry, Kyunghyun Cho and Yoshua Bengio (2014), *Neural Machine Translation by Jointly Learning to Align and Translate*, arXiv: 1409.0473 [cs.CL].

Baroni, Marco and Alessandro Lenci (2010), "Distributional Memory: A General Framework for Corpus-Based Semantics", *in*: *Computational Linguistics* **36**.*4*, pp. 673–721, URL: https://www.aclweb.org/anthology/J10-4006.

Bojanowski, Piotr, Edouard Grave, Armand Joulin and Tomas Mikolov (2017), "Enriching Word Vectors with Subword Information", *in*: *Transactions of the Association for Computational Linguistics* **5**, pp. 135–146, ISSN: 2307-387X.

Botha, Jan and Phil Blunsom (2014), "Compositional Morphology for Word Representations and Language Modelling", *in*: *Proceedings of the 31st International Conference on Machine*

Learning (ICML'14), vol. 32, 2, pp. 1899–1907, URL: http://proceedings.mlr.press/v32/botha14.html.

Bouamor, Dhouha, Nasredine Semmar and Pierre Zweigenbaum (2013), "Context Vector Disambiguation for Bilingual Lexicon Extraction from Comparable Corpora", *in*: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, Sofia, Bulgaria, pp. 759–764, URL: http://www.aclweb.org/anthology/P13-2133.

Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger and Roopak Shah (1993), "Signature Verification using a "Siames" Time Delay Neural Network", *in*: *International Journal of Pattern Recognition and Artificial Intelligence* **7**.*4*, pp. 669–688.

Bullinaria, John A. and Joseph P. Levy (2007), "Extracting semantic representations from word co-occurrence statistics: A computational study", *in*: *Behavior Research Methods* **39**.*3*, pp. 510–526.

Camacho-Collados, José, Mohammad Taher Pilehvar and Roberto Navigli (2016), "NASARI: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities", *in*: *Artificial Intelligence* **240**, pp. 36–64, ISSN: 0004-3702, URL: http://www.sciencedirect.com/science/article/pii/S0004370216300820.

Chen, Yun, Yang Liu, Yong Cheng and Victor O.K. Li (2017), "A Teacher-Student Framework for Zero-Resource Neural Machine Translation", *in*: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*, Vancouver, Canada, pp. 1925–1935, DOI: 10.18653/v1/P17-1176, URL: http://aclweb.org/anthology/P17-1176.

Chiao, Yun-Chuang and Pierre Zweigenbaum (2002), "Looking for Candidate Translational Equivalents in Specialized, Comparable Corpora", *in*: *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, Stroudsburg, PA, USA, pp. 1–5, DOI: 10.3115/1071884.1071904, URL: https://doi.org/10.3115/1071884.1071904.

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio (2014), "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", *in*: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, Doha, Qatar, pp. 1724–1734, DOI: 10.3115/v1/D14-1179, URL: http://aclweb.org/anthology/D14-1179.

Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho and Yoshua Bengio (2014), *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*, arXiv: 1412.3555 [cs.NE].

Church, Kenneth Ward and Patrick Hanks (1990), "Word Association Norms, Mutual Information, and Lexicography", *in*: *Computational Linguistics* **16**.*1*, pp. 22–29, ISSN: 0891-2017, URL: http://dl.acm.org/citation.cfm?id=89086.89095.

Dagan, Ido, Fernando Pereira and Lillian Lee (1994), "Similarity-based Estimation of Word Cooccurrence Probabilities", *in*: *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics (ACL '94)*, Las Cruces, New Mexico, pp. 272–278, DOI: 10.3115/981732.981770, URL: https://doi.org/10.3115/981732.981770.

Dai, Andrew M and Quoc V Le (2015), "Semi-supervised Sequence Learning", *in*: *Advances in Neural Information Processing Systems 28 (NIPS'15)*, pp. 3079–3087, URL: http://papers.nips.cc/paper/5949-semi-supervised-sequence-learning.pdf.

Dai, Zihang, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le and Ruslan Salakhutdinov (2019), *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*, arXiv: 1901.02860 [cs.LG].

Das, Arpita, Harish Yenala, Manoj Chinnakotla and Manish Shrivastava (2016), "Together we stand: Siamese Networks for Similar Question Retrieval", *in*: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*, Berlin, Germany, pp. 378–387, DOI: 10.18653/v1/P16-1036, URL: http://aclweb.org/anthology/P16-1036.

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer and Richard Harshman (1990), "Indexing by latent semantic analysis", *in*: *Journal of the Association for Information Science and Technology* **41**.*6*, pp. 391–407.

Del, Maksym, Andre Tättar and Mark Fishel (2018), "Phrase-based Unsupervised Machine Translation with Compositional Phrase Embeddings", *in*: *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, Belgium, Brussels: Association for Computational Linguistics, pp. 361–367, URL: https://www.aclweb.org/anthology/W18-6407.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova (2018), *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv: 1810.04805 [cs.CL].

Dunning, Ted (1993), "Accurate Methods for the Statistics of Surprise and Coincidence", *in*: *Computational Linguistics* **19**.*1*, pp. 61–74, URL: https://www.aclweb.org/anthology/J93-1003.

Elman, Jeffrey L. (1990), "Finding Structure in Time", *in*: *Cognitive Science* **14**.*2*, pp. 179–211.

Evert, Stefan (2005), "The Statistics of Word Cooccurrences: Word Pairs and Collocations", PhD thesis, Universität Stuttgart.

Fano, Robert M. (1961), *Transmission of Information: A Statistical Theory of Communications*, Cambridge, MA, USA: MIT Press.

Faruqui, Manaal and Chris Dyer (2014), "Improving Vector Space Word Representations Using Multilingual Correlation", *in*: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL'14)*, Gothenburg, Sweden, pp. 462–471, URL: http://www.aclweb.org/anthology/E14-1049.

Finch, Geoffrey (2000), *Linguistic Terms and Concepts*, London: Macmillan Education UK, ISBN: 978-1-349-27748-3, DOI: 10.1007/978-1-349-27748-3_5, URL: https://doi.org/10.1007/978-1-349-27748-3_5.

Firat, Orhan, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman Vural and Kyunghyun Cho (2016), "Zero-Resource Translation with Multi-Lingual Neural Machine Translation", *in*: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*, Austin, TeX, USA, pp. 268–277, DOI: 10.18653/v1/D16-1026, URL: http://aclweb.org/anthology/D16-1026.

Fukushima, Kunihiko (1980), "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *in*: *Biological Cybernetics* **36**.*4*, pp. 193–202, DOI: 10.1007/BF00344251, URL: https://doi.org/10.1007/BF00344251.

Fung, Pascale (1995), "Compiling Bilingual Lexicon Entries From a non-Parallel English-Chinese Corpus", *in*: *Proceedings of the 3rd Annual Workshop on Very Large Corpora (VLC'95)*, Cambridge, MA, USA, pp. 173–183.

Garten, Justin, Kenji Sagae, Volkan Ustun and Morteza Dehghani (2015), "Combining Distributed Vector Representations for Words", *in*: *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, Denver, CO, USA, pp. 95–101, DOI: 10.3115/v1/W15-1513, URL: https://www.aclweb.org/anthology/W15-1513.

Gers, Felix A., Jürgen A. Schmidhuber and Fred A. Cummins (2000), "Learning to Forget: Continual Prediction with LSTM", *in*: *Neural Computation* **12**.*10*, pp. 2451–2471, ISSN:

0899-7667, DOI: 10.1162/089976600300015015, URL: http://dx.doi.org/10.1162/089976600300015015.

Glorot, Xavier, Antoine Bordes and Yoshua Bengio (2011), "Deep Sparse Rectifier Neural Networks", *in*: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS'11)*, ed. by Geoffrey Gordon, David Dunson and Miroslav Dudík, vol. 15, Proceedings of Machine Learning Research, Fort Lauderdale, FL, USA: PMLR, pp. 315–323, URL: http://proceedings.mlr.press/v15/glorot11a.html.

Goikoetxea, Josu, Eneko Agirre and Aitor Soroa (2016), "Single or Multiple? Combining Word Representations Independently Learned from Text and WordNet", *in*: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*, Phoenix, AZ, USA, pp. 2608–2614, URL: http://dl.acm.org/citation.cfm?id=3016100.3016266.

Goldberg, Yoav and Omer Levy (2014), *word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method*, arXiv: 1402.3722 [cs.CL].

Goller, C. and A. Küchler (1996), "Learning Task-Dependent Distributed Representations by Backpropagation Through Structure", *in*: *Proceedings of International Conference on Neural Networks (ICNN'96)*, Washington, DC, USA, pp. 347–352, URL: https://ieeexplore.ieee.org/document/548916.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio (2014), "Generative Adversarial Nets", *in*: *Advances in Neural Information Processing Systems 27 (NIPS'14)*, vol. 2, Montreal, Canada, pp. 2672–2680, URL: http://dl.acm.org/citation.cfm?id=2969033.2969125.

Gouws, Stephan, Yoshua Bengio and Greg Corrado (2015), "BilBOWA: Fast Bilingual Distributed Representations Without Word Alignments", *in*: *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML'15)*, Lille, France, pp. 748–756, URL: http://dl.acm.org/citation.cfm?id=3045118.3045199.

Grefenstette, Gregory (1999), "The World Wide Web as a Resource for Example-Based Machine Translation Tasks", *in*: *Proceedings of the ASLIB Conference on Translating and the Computer 21*, London, UK.

Greff, Klaus, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink and Jurgen Schmidhuber (2017), "LSTM: A Search Space Odyssey", *in*: *IEEE Transactions on Neural Networks and Learning Systems* **28**.*10*, pp. 2222–2232, ISSN: 2162-2388, DOI: 10.1109/

tnnls.2016.2582924, URL: http://dx.doi.org/10.1109/TNNLS.2016.2582924.

Groc, Clément De (2011), "Babouk: Focused Web Crawling for Corpus Compilation and Automatic Terminology Extraction", *in*: *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 1, pp. 497–498, DOI: 10.1109/WI-IAT.2011.253.

Harris, Zellig (1954), "Distributional Structure", *in*: *Word* **10**.*2–3*, pp. 146–162.

Hazem, Amir and Béatrice Daille (2018), "Word Embedding Approach for Synonym Extraction of Multi-Word Terms", *in*: *Proceedings of the 11th edition of the Language Resources and Evaluation Conference (LREC'18)*, Miyazaki, Japan, pp. 297–303, URL: http://www.lrec-conf.org/proceedings/lrec2018/pdf/36.pdf.

Hazem, Amir and Emmanuel Morin (2016), "Efficient Data Selection for Bilingual Terminology Extraction from Comparable Corpora", *in*: *Proceedings of the 26th International Conference on Computational Linguistics (COLING'16)*, Osaka, Japan, pp. 3401–3411, URL: http://aclweb.org/anthology/C16-1321.

— (2017), "Bilingual Word Embeddings for Bilingual Terminology Extraction from Specialized Comparable Corpora", *in*: *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP'17)*, Taipei, Taiwan, pp. 685–693, URL: http://aclweb.org/anthology/I17-1069.

He, Di, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu and Wei-Ying Ma (2016), "Dual Learning for Machine Translation", *in*: *Advances in Neural Information Processing Systems 29 (NIPS'16)*, pp. 820–828, URL: http://papers.nips.cc/paper/6469-dual-learning-for-machine-translation.pdf.

Hochreiter, Sepp and Jürgen Schmidhuber (1997), "Long Short-Term Memory", *in*: *Neural Computation* **9**.*8*, pp. 1735–1780, ISSN: 0899-7667, DOI: 10.1162/neco.1997.9.8.1735, URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hubel, David H. and Torsten Nils Wiesel (1968), "Receptive fields and functional architecture of monkey striate cortex", *in*: *The Journal of Physiology* **195**.*1*, pp. 215–243.

Irsoy, Ozan and Claire Cardie (2014), "Deep Recursive Neural Networks for Compositionality in Language", *in*: *Advances in Neural Information Processing Systems 27 (NIPS'14)*, pp. 2096–2104, URL: http://papers.nips.cc/paper/5551-deep-recursive-neural-networks-for-compositionality-in-language.pdf.

Jakubina, Laurent and Phillippe Langlais (2017), "Reranking Translation Candidates Produced by Several Bilingual Word Similarity Sources", *in*: *Proceedings of the 15th Conference of*

the European Chapter of the Association for Computational Linguistics (EACL'17), Valencia, Spain, pp. 605–611, URL: http://www.aclweb.org/anthology/E17-2096.

Johnson, Melvin et al. (2017), "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation", *in*: *Transactions of the Association for Computational Linguistics* **5**, pp. 339–351, URL: http://aclweb.org/anthology/Q17-1024.

Jozefowicz, Rafal, Wojciech Zaremba and Ilya Sutskever (2015), "An Empirical Exploration of Recurrent Network Architectures", *in*: *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML'15)*, vol. 37, Lille, France, pp. 2342–2350, URL: http://dl.acm.org/citation.cfm?id=3045118.3045367.

Keenan, Edward L. and Leonard M. Faltz (1985), *Boolean Semantics for Natural Language*, D. Reidel Publishing Company.

Kenter, Tom, Alexey Borisov and Maarten de Rijke (2016), "Siamese CBOW: Optimizing Word Embeddings for Sentence Representations", *in*: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*, Berlin, Germany, pp. 941–951, DOI: 10.18653/v1/P16-1089, URL: http://aclweb.org/anthology/P16-1089.

Kim, Yoon, Yacine Jernite, David Sontag and Alexander M. Rush (2016), "Character-aware Neural Language Models", *in*: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*, Phoenix, Arizona, pp. 2741–2749, URL: http://dl.acm.org/citation.cfm?id=3016100.3016285.

Koch, Gregory, Richard Zemel and Ruslan Salakhutdinov (2015), "Siamese Neural Networks for One-shot Image Recognition", *in*: *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*, Lille, France.

Kotlerman, Lili, Ido Dagan, Idan Szpektor and Maayan Zhitomirsky-Geffet (2010), "Directional Distributional Similarity for Lexical Inference", *in*: *Natural Language Engineering* **16**.*4*, pp. 359–389, ISSN: 1351-3249, DOI: 10.1017/S1351324910000124, URL: http://dx.doi.org/10.1017/S1351324910000124.

Kroeger, P.R. (2005), *Analyzing Grammar: An Introduction*, Cambridge University Press, ISBN: 9781139443517, URL: https://books.google.fr/books?id=rSglHbBaNyAC.

Lample, Guillaume and Alexis Conneau (2019), *Cross-lingual Language Model Pretraining*, arXiv: 1901.07291 [cs.CL].

Lample, Guillaume, Alexis Conneau, Ludovic Denoyer and Marc'Aurelio Ranzato (2018), "Unsupervised Machine Translation Using Monolingual Corpora Only", *in*: *Proceedings*

of the 6th International Conference on Learning Representations (ICLR'18), Vancouver, Canada.

Lazaridou, Angeliki, Georgiana Dinu and Marco Baroni (2015), "Hubness and Pollution: Delving into Cross-Space Mapping for Zero-Shot Learning", *in*: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP'15)*, Beijing, China, pp. 270–280, DOI: 10.3115/v1/P15-1027, URL: http://aclweb.org/anthology/P15-1027.

Lazaridou, Angeliki, Marco Marelli, Roberto Zamparelli and Marco Baroni (2013), "Compositionally Derived Representations of Morphologically Complex Words in Distributional Semantics", *in*: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, vol. 1, Sofia, Bulgaria, pp. 1517–1526, URL: https://www.aclweb.org/anthology/P13-1149.

Le, Phong and Willem Zuidema (2014), "The Inside-Outside Recursive Neural Network model for Dependency Parsing", *in*: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, Doha, Qatar, pp. 729–739, DOI: 10.3115/v1/D14-1081, URL: http://aclweb.org/anthology/D14-1081.

Lebret, Rémi and Ronan Collobert (2014), "Word Embeddings through Hellinger PCA", *in*: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL'14)*, Gothenburg, Sweden, pp. 482–490, DOI: 10.3115/v1/E14-1051, URL: https://www.aclweb.org/anthology/E14-1051.

Lecun, Yann, Léon Bottou, Yoshua Bengio and Patrick Haffner (1998), "Gradient-based learning applied to document recognition", *in*: *Proceedings of the IEEE* **86**.*11*, pp. 2278–2324.

Lin, Dekang (1998), "Automatic Retrieval and Clustering of Similar Words", *in*: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL-COLING '98)*, vol. 2, Montreal, Quebec, Canada, pp. 768–774, DOI: 10.3115/980691.980696, URL: https://doi.org/10.3115/980691.980696.

Ling, Wang, Yulia Tsvetkov, Silvio Amir, Ramon Fermandez, Chris Dyer, Alan W Black, Isabel Trancoso and Chu-Cheng Lin (2015), "Not All Contexts Are Created Equal: Better Word Representations with Variable Attention", *in*: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*, Lisbon, Portugal, pp. 1367–1372, DOI: 10.18653/v1/D15-1161, URL: http://aclweb.org/anthology/D15-1161.

Liu, Jingshu, Emmanuel Morin and Sebastián Peña Saldarriaga (2018), "Towards a unified framework for bilingual terminology extraction of single-word and multi-word terms", *in*: *Proceedings of the 27th International Conference on Computational Linguistics (COLING'18)*, Santa Fe, NM, USA, pp. 2855–2866, URL: http://aclweb.org/anthology/C18-1242.

Luong, Minh-Thang and Christopher D. Manning (2016), "Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models", *in*: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*, vol. 1, Berlin, Germany, pp. 1054–1063, DOI: 10.18653/v1/P16-1100, URL: https://www.aclweb.org/anthology/P16-1100.

Luong, Minh-Thang, Hieu Pham and Christopher D. Manning (2015a), "Bilingual Word Representations with Monolingual Quality in Mind", *in*: *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, Denver, Colorado: Association for Computational Linguistics, pp. 151–159, DOI: 10.3115/v1/W15-1521, URL: https://www.aclweb.org/anthology/W15-1521.

— (2015b), "Effective Approaches to Attention-based Neural Machine Translation", *in*: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*, Lisbon, Portugal, pp. 1412–1421, DOI: 10.18653/v1/D15-1166, URL: http://aclweb.org/anthology/D15-1166.

Manning, Christoper D., Prabhakar Raghavan and Hinrich Schütze (2008), *An Introduction to Information Retrieval*, Cambridge University Press, DOI: 10.1017/CBO9780511809071.

Martin, Kyle, Nirmalie Wiratunga, Sadiq Sani, Stewart Massie and Jérémie Clos (2017), "A Convolutional Siamese Network for Developing Similarity Knowledge in the SelfBACK Dataset", *in*: *In Proceedings of the Case-Based Reasoning and Deep Learning Workshop (CBRDL'17)*, Trondheim, Norway, pp. 85–94, URL: http://ceur-ws.org/Vol-2028/paper8.pdf.

McCulloch, Warren S. and Walter Pitts (1943), "A logical calculus of the ideas immanent in nervous activity", *in*: *The bulletin of Mathematical Biophysics* **5**.*4*, pp. 115–133, URL: https://doi.org/10.1007/BF02478259.

Mikolov, Tomas, Kai Chen, Greg Corrado and Jeffrey Dean (2013), *Efficient Estimation of Word Representations in Vector Space*, arXiv: 1301.3781 [cs.CL].

Mikolov, Tomas, Quoc V. Le and Ilya Sutskever (2013), *Exploiting Similarities among Languages for Machine Translation*, arXiv: 1309.4168 [cs.CL].

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado and Jeff Dean (2013), "Distributed Representations of Words and Phrases and their Compositionality", *in*: *Advances Neural Information Processing Systems 26 (NIPS'13)*, pp. 3111–3119, URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Mnih, Andriy and Koray Kavukcuoglu (2013), "Learning Word Embeddings Efficiently with Noise-contrastive Estimation", *in*: *Advances Neural Information Processing Systems 26 (NIPS'13)*, vol. 2, Lake Tahoe, Nevada, pp. 2265–2273, URL: http://dl.acm.org/citation.cfm?id=2999792.2999865.

Mogadala, Aditya and Achim Rettinger (2016), "Bilingual Word Embeddings from Parallel and Non-parallel Corpora for Cross-Language Text Classification", *in*: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'16)*, San Diego, California, pp. 692–702, DOI: 10.18653/v1/N16-1083, URL: https://www.aclweb.org/anthology/N16-1083.

Moore, Robert C. and William Lewis (2010), "Intelligent Selection of Language Model Training Data", *in*: *Proceedings of the ACL 2010 Conference Short Papers*, pp. 220–224.

Morin, Emmanuel and Béatrice Daille (2012), "Revising the Compositional Method for Terminology Acquisition from Comparable Corpora", *in*: *Proceedings of the 24rd International Conference on Computational Linguistics (COLING'12)*, Mumbai, India, pp. 1797–1810, URL: http://www.aclweb.org/anthology/C12-1110.

Niwa, Yoshiki and Yoshihiko Nitta (1994), "Co-occurrence Vectors from Corpora vs. Distance Vectors from Dictionaries", *in*: *Proceedings of the 15th Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan, pp. 304–309, DOI: 10.3115/991886.991938, URL: https://doi.org/10.3115/991886.991938.

Paulus, Romain, Richard Socher and Christopher D Manning (2014), "Global Belief Recursive Neural Networks", *in*: *Advances in Neural Information Processing Systems 27 (NIPS'14)*, pp. 2888–2896, URL: http://papers.nips.cc/paper/5275-global-belief-recursive-neural-networks.pdf.

Pennington, Jeffrey, Richard Socher and Christopher Manning (2014), "GloVe: Global Vectors for Word Representation", *in*: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, Doha, Qatar, pp. 1532–1543.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer (2018), "Deep Contextualized Word Representations", *in*: *Proceed-*

ings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'18), New Orleans, LA, USA, pp. 2227–2237, DOI: 10.18653/v1/N18-1202, URL: http://aclweb.org/anthology/N18-1202.

Qiu, Siyu, Qing Cui, Jiang Bian, Bin Gao and Tie-Yan Liu (2014), "Co-learning of Word Representations and Morpheme Representations", *in*: *Proceedings the 25th International Conference on Computational Linguistics: Technical Papers (COLING'14)*, Dublin, Ireland, pp. 141–150, URL: https://www.aclweb.org/anthology/C14-1015.

Radford, Alec, Jeff Wu, Rewon Child, David Luan, Dario Amodei and Ilya Sutskever (2019), *Language Models are Unsupervised Multitask Learners*, https://github.com/openai/gpt-2, URL: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

Rapp, Reinhard (1999), "Automatic Identification of Word Translations from Unrelated English and German Corpora", *in*: *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, College Park, Maryland, USA, pp. 519–526, DOI: 10.3115/1034678.1034756, URL: http://www.aclweb.org/anthology/P99-1067.

Robitaille, Xavier, Yasuhiro Sasaki, Masatsugu Tonoike, Satoshi Sato and Takehito Utsuro (2006), "Compiling French-Japanese Terminologies from the Web", *in*: *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, Trento, Italy, pp. 225–232, URL: http://aclweb.org/anthology/E06-1029.

Rosenblatt, F. (1958), "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain", *in*: *Psychological Review* **65**.*6*, pp. 386–408.

Saha, Amrita, Mitesh M. Khapra, Sarath Chandar, Janarthanan Rajendran and Kyunghyun Cho (2016), "A Correlational Encoder Decoder Architecture for Pivot Based Sequence Generation", *in*: *Proceedings of the 26th International Conference on Computational Linguistics (COLING'16)*, Osaka, Japan, pp. 109–118, URL: http://aclweb.org/anthology/C16-1011.

Schluter, Natalie (2018), "The Word Analogy Testing Caveat", *in*: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'18)*, New Orleans, LA, USA, pp. 242–246, DOI: 10.18653/v1/N18-2039, URL: http://aclweb.org/anthology/N18-2039.

Schuster, Tal, Ori Ram, Regina Barzilay and Amir Globerson (2019), "Cross-Lingual Alignment of Contextual Word Embeddings, with Applications to Zero-shot Dependency Parsing", *in*: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, pp. 1599–1613, URL: https://www.aclweb.org/anthology/N19-1162.

Sennrich, Rico, Barry Haddow and Alexandra Birch (2016a), "Improving Neural Machine Translation Models with Monolingual Data", *in*: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*, Berlin, Germany, pp. 86–96, DOI: 10.18653/v1/P16-1009, URL: http://aclweb.org/anthology/P16-1009.

— (2016b), "Neural Machine Translation of Rare Words with Subword Units", *in*: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*, Berlin, Germany, pp. 1715–1725, DOI: 10.18653/v1/P16-1162, URL: https://www.aclweb.org/anthology/P16-1162.

Shannon, Claude E and Warren Weaver (1949), *The mathematical theory of communication*, Urbana, Illinois, USA: University of Illinois Press.

Shigeto, Yutaro, Ikumi Suzuki, Kazuo Hara, Masashi Shimbo and Yuji Matsumoto (2015), "Ridge Regression, Hubness, and Zero-Shot Learning", *in*: *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD'15)*, ed. by Annalisa Appice, Pedro Pereira Rodrigues, Vítor Santos Costa, Carlos Soares, João Gama and Alípio Jorge, Porto, Portugal, pp. 135–151.

Smith, Samuel L., David H. P. Turban, Steven Hamblin and Nils Y. Hammerla (2017), "Offline bilingual word vectors, orthogonal transformations and the inverted softmax", *in*: *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*, Toulon, France.

Sobin, N. (2010), *Syntactic Analysis: The Basics*, Wiley, ISBN: 9781444390704, URL: https://books.google.fr/books?id=X0lSCdzzJOsC.

Socher, Richard, John Bauer, Christopher D. Manning and Ng Andrew Y. (2013), "Parsing with Compositional Vector Grammars", *in*: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, Sofia, Bulgaria, pp. 455–465, URL: http://aclweb.org/anthology/P13-1045.

Socher, Richard, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng and Christopher Potts (2013), "Recursive Deep Models for Semantic Compositionality

Over a Sentiment Treebank", *in*: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, Seattle, WA, USA, pp. 1631–1642, URL: http://aclweb.org/anthology/D13-1170.

Søgaard, Anders, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet and Anders Johannsen (2015), "Inverted indexing for cross-lingual NLP", *in*: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP'15)*, vol. 1, Beijing, China, pp. 1713–1722, DOI: 10.3115/v1/P15-1165, URL: https://www.aclweb.org/anthology/P15-1165.

Sutskever, Ilya, Oriol Vinyals and Quoc V Le (2014), "Sequence to Sequence Learning with Neural Networks", *in*: *Advances in Neural Information Processing Systems 27 (NIPS'14)*, pp. 3104–3112, URL: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Szabó, Zoltán Gendler (2017), "Compositionality", *in*: *The Stanford Encyclopedia of Philosophy*, ed. by Edward N. Zalta, Summer 2017, Metaphysics Research Lab, Stanford University.

Tanaka, Takaaki (2002), "Measuring the Similarity Between Compound Nouns in Different Languages Using Non-parallel Corpora", *in*: *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, Taipei, Taiwan, pp. 1–7, URL: https://doi.org/10.3115/1072228.1072293.

Turney, Peter (2001), "Mining the web for synonyms: PMI-IR versus LSA on TOEFL", *in*: *Proceedings of the Twelfth European Conference on Machine Learning (ECML'01)*, pp. 491–502, DOI: 10.1007/3-540-44795-4_42.

Turney, Peter D. and Patrick Pantel (2010), "From Frequency to Meaning: Vector Space Models of Semantics", *in*: *Journal of Artificial Intelligence Research* **37**.*1*, pp. 141–188, ISSN: 1076-9757, URL: http://dl.acm.org/citation.cfm?id=1861751.1861756.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser and Illia Polosukhin (2017), "Attention is All you Need", *in*: *Advances in Neural Information Processing Systems 30 (NIPS'17)*, pp. 5998–6008, URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio and Pierre-Antoine Manzagol (2008), "Extracting and Composing Robust Features with Denoising Autoencoders", *in*: *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*, Helsinki, Finland,

pp. 1096–1103, ISBN: 978-1-60558-205-4, DOI: 10.1145/1390156.1390294, URL: http://doi.acm.org/10.1145/1390156.1390294.

Vulic, Ivan and Marie-Francine Moens (2016), "Bilingual Distributed Word Representations from Document-aligned Comparable Data", *in*: *Journal of Artificial Intelligence Research* **55**.*1*, pp. 953–994, ISSN: 1076-9757, URL: http://dl.acm.org/citation.cfm?id=3013558.3013583.

Wang, Liwei, Yin Li and Svetlana Lazebnik (2016), "Learning Deep Structure-Preserving Image-Text Embeddings", *in*: *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*, Las Vegas, NV, USA, pp. 5005–5013, DOI: 10.1109/CVPR.2016.541.

Wang, Longyue, Derek F. Wong, Lidia S. Chao, Yi Lu and Junwen Xing (2014), "A Systematic Comparison of Data Selection Criteria for SMT Domain Adaptation", *in*: *The Scientific World Journal* **2014**, p. 10, DOI: 10.1155/2014/745485.

Werbos, Paul J (1990), "Backpropagation through time: what it does and how to do it", *in*: *Proceedings of the IEEE* **78**.*10*, pp. 1550–1560, DOI: 10.1109/5.58337.

Wieting, John, Mohit Bansal, Kevin Gimpel and Karen Livescu (2016), "Charagram: Embedding Words and Sentences via Character n-grams", *in*: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*, Austin, Texas, pp. 1504–1515, DOI: 10.18653/v1/D16-1157, URL: https://www.aclweb.org/anthology/D16-1157.

Williams, Ronald J. and David Zipser (1995), "Backpropagation: Theory, architectures, and applications", *in*: ed. by Yves Chauvin and David E. Rumelhart, Hillsdale, NJ, USA: Lawrence Erlbaum Associates, Inc, chap. Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity, pp. 433–486.

Wu, Jiawei, Xin Wang and William Yang Wang (2019), "Extract and Edit: An Alternative to Back-Translation for Unsupervised Neural Machine Translation", *in*: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'19)*, Minneapolis, Minnesota, pp. 1173–1183, URL: https://www.aclweb.org/anthology/N19-1120.

Xing, Chao, Dong Wang, Chao Liu and Yiye Lin (2015), "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 (NAACL'15)*, Denver, CO, USA, pp. 1006–1011, URL: http://www.aclweb.org/anthology/N15-1104.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel and Yoshua Bengio (2015), "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", *in*: *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*, vol. 37, Lille, France, pp. 2048–2057, URL: http://proceedings.mlr.press/v37/xuc15.html.

Yang, Zhen, Wei Chen, Feng Wang and Bo Xu (2018), "Unsupervised Neural Machine Translation with Weight Sharing", *in*: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL'18)*, Melbourne, Australia, pp. 46–55, URL: http://aclweb.org/anthology/P18-1005.

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov and Quoc V. Le (2019), *XLNet: Generalized Autoregressive Pretraining for Language Understanding*, arXiv: 1906.08237 [cs.CL].

Zagoruyko, Sergey and Nikos Komodakis (2015), "Learning to Compare Image Patches via Convolutional Neural Networks", *in*: *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, Boston, MA, USA, pp. 885–894, DOI: 10.1109/CVPR.2015.7299064, URL: https://hal.archives-ouvertes.fr/hal-01246261.

Zhang, Jiajun and Chengqing Zong (2016), "Exploiting Source-side Monolingual Data in Neural Machine Translation", *in*: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*, Austin, TX, USA, pp. 1535–1545, DOI: 10.18653/v1/D16-1160, URL: http://aclweb.org/anthology/D16-1160.

Zhang, Yuan, David Gaddy, Regina Barzilay and Tommi Jaakkola (2016), "Ten Pairs to Tag – Multilingual POS Tagging via Coarse Mapping between Embeddings", *in*: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'16)*, San Diego, CA, USA, pp. 1307–1317, DOI: 10.18653/v1/N16-1156, URL: http://aclweb.org/anthology/N16-1156.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE NANTES

Titre: Apprentissage de représentations cross-lingue d'expressions de longueur variable

Mot clés : plongement lexical bilingue, alignement d'expressions, apprentissage non-supervisé

Resumé : L'étude de l'extraction de lexiques bilingues à partir de corpus comparables a été souvent circonscrite aux mots simples. Les méthodes classiques ne peuvent gérer les expressions complexes que si elles sont de longueur identique, tandis que les méthodes de plongements de mots modélisent les expressions comme une seule unité. Ces dernières nécessitent beaucoup de données, et ne peuvent pas gérer les expressions hors vocabulaire. Dans cette thèse, nous nous intéressons à la modélisation d'expressions de longueur variable par co-occurrences et par les méthodes neuronales état de l'art. Nous étudions aussi l'apprentissage de représentation d'expressions supervisé et non-supervisé. Nous proposons deux contributions majeures. Premièrement, une nouvelle architecture appelée *tree-free recursive neural network* (TF-RNN) pour la modélisation d'expressions indépendamment de leur longueur. En apprenant à prédire le contexte de l'expression à partir de son vecteur encodé, nous surpassons les systèmes état de l'art de synonymie monolingue en utilisant seulement le texte brut pour l'entraînement. Deuxièmement, pour la modélisation cross-lingue, nous incorporons une architecture dérivée de TF-RNN dans un modèle *encodeur-décodeur* avec un mécanisme de pseudo contre-traduction inspiré de travaux sur la traduction automatique neurale non-supervisée. Notre système améliore significativement l'alignement bilingue des expressions de longueurs différentes.

Title: Unsupervised cross-lingual representation modeling for variable length phrases

Keywords: bilingual word embedding, bilingual phrase alignment, unsupervised learning

Abstract: Significant advances have been achieved in bilingual word-level alignment from comparable corpora, yet the challenge remains for phrase-level alignment. Traditional methods to phrase alignment can only handle phrase of equal length, while word embedding based approaches learn phrase embeddings as individual vocabulary entries suffer from the data sparsity and cannot handle out of vocabulary phrases. Since bilingual alignment is a vector comparison task, phrase representation plays a key role. In this thesis, we study the approaches for unified phrase modeling and cross-lingual phrase alignment, ranging from co-occurrence models to most recent neural state-of-the-art approaches. We review supervised and unsupervised frameworks for modeling cross-lingual phrase representations. Two contributions are proposed in this work. First, a new architecture called *tree-free recursive neural network* (TF-RNN) for modeling phrases of variable length which, combined with a wrapped context prediction training objective, outperforms the state-of-the-art approaches on monolingual phrase synonymy task with only plain text training data. Second, for cross-lingual modeling, we propose to incorporate an architecture derived from TF-RNN in an *encoder-decoder* model with a pseudo back translation mechanism inspired by unsupervised neural machine translation. Our proposition improves significantly bilingual alignment of different length phrases.