N° d'ordre :
N° de série :

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research

ECHAHID HAMMA LAKHDAR UNIVERSITY - EL OUED
FACULTY OF EXACT SCIENCES
Computer Science department

End of Study Memory
Presented for the Diploma of

ACADEMIC MASTER

Domain : **Mathematics and Computer Science**
spinneret: **Computer Science**
Speciality : **Artificial Intelligence and Distributed Systems**

Presented by :

- **Raki Lachraf**
- **Youcef Ayachi**

Theme

Cross-Lingual Semantic Textual Similarity for English and Arabic Sentences

Supported in: - - 2019 In front of jury:

M.	……	MCA	President
M.	……	MAA	Reporter
Dr.	El Moataz Billah Nagoudi	MAA	Supervisor

University Year: 2018/2019

Acknowledgements

First of all, we thank ***Allah*** for giving us the opportunity to be at this position and providing us with knowledge and patience to finish this work. Second, we would like to express our sincere gratitude to Dr. *Nagoudi El Moatez Billah*, our supervisor for his continued support for research, inquiry, patience, motivation and knowledge. As well as his help and guidance and valuable advice at all time of research and throughout these months of writing this thesis.

In addition to our supervisor, We would like to thank our parents, the friends we had their help especially *Aissa Trad* and *Cherif Bali*. We thank the rest of the discussion committee for their insightful comments, encouragement and advice, and anyone who has had a hand from near or far in helping us even by encouraging and motivating.

We cheer this work with our friend ... brother in the foreign land *Affoun Abdelhay* wishing for him but success and happiness.

Abstract

Artificial Intelligence is a computer science sub-field that had many lights focused on it these last three decades. Computational Linguistics, also widely known as Natural Language Processing (NLP), is the domain that covers such studies that correspond to human's most famous way of communication. Natural Language Processing treats many major problems starting from Sentiment Analysis, Plagiarism Detection, Machine Translation down to Information Retrieval, Keywords Extraction and Semantic Textual Similarity which are considered as keystones for solving any other high level Natural Language Processing problem. Particularly, Semantic Textual Similarity can be put as a blueprint head start in many Natural Language Processing projects. Not just in a mono-lingual aspect but also multilingual. In this work we proposed a Cross-Lingual Semantic Similarity System concerning English and Arabic. We collected and preprocessed a large textual dataset. While achieving this purpose we stepped into a crucial decision of choosing one efficient system core known as the Word Embedding Model. As a matter of fact we have built our own and carried on towards our Similarity System. Eventually, we combined the system with three different weighting methods looking for more improvement.

Keywords: Natural Language Processing, Word Embeddings, Machine Translation, Semantic Similarity.

Resumé

L'intelligence artificielle est un sous-domaine de l'informatique qui a suscit de nombreuses études et intra ses trois dernières décennies. Le Traitement Automatique du Langage Naturel (TALN), appelé aussi Traitement Automatique des Langues (TAL), est parmi les domaines qui abord ces études. La TAL traite de nombreux problèmes majeurs allant de l'analyse des sentiments, la détection de plagiat, la traduction automatique à la recherche d'informations, l'extraction de mots-clés et la similarité sémantique textuelle, qui sont considérés comme des problèmes de references pour la résolution de tout autre problème de traitement du langage naturel. En particulier, la similarité textuelle sémantique peut être considérée comme une base de départ dans de nombreux projets de traitement du langage naturel. Pas seulement dans un aspect monolingue, mais aussi multilingue. Dans ce travail, nous avons proposé un système de similarité sémantique multilingue en anglais et en arabe. Pour atteint cet objectif, nous avons commencé par la collection et pré traitement d'un corpus parallèle Anglais-Arabe. Ensuite, nous avons construit notre propre système en se basant sur un modèle d'incorporation de mots. Finalement, nous avons combiné le système avec trois méthodes de pondération différentes afin d'obtenir une amélioration supplémentaires.

Mots-clés: traitement automatique du langage naturel, incorporation de mots, traduction automatique, similarité sémantique.

ملـخّـــص

الذكاء الاصطناعي هو حقل فرعي لعلوم الكمبيوتر كان له حظ وافر من الأضواء المسلطة عليه في العقود الثلاث المنصرمة. تحديدا دراسة الذكاء المتمثل في طريقة التحاور البشرية. إن المعالجة الطبيعية للغة أو ما يسمى اللغويات الحاسوبية هي في الغالب تغطي مثل هذه الدراسات. و يعالج مجال اللغويات الحاسوبية العديد من المشكلات الرئيسية بدءاً من تحليل المعنويات و المشاعر ، الكشف عن الانتحال الترجمة الآلية ابتداءا من استخراج المعلومات ، استخراج الكلمات الأساسية و أخيرا التشابه النصي الدلالي ، والتي تُعتبر حجر الأساس لحل أي مشكل آخر ذو درجة تعقيد عالية على مستوى معالجة اللغة الطبيعية. على وجه الخصوص ، يمكن وضع التشابه النصي الدلالي كجزء أساسي من أي مخطط رئيسي للشروع في العديد من مشاريع معالجة اللغة الطبيعية. ليس فقط في جانب أحادي اللغة ولكن أيضًا متعدد اللغات. في هذا العمل ، اقترحنا نظامًا للتشابه الدلالي عبر اللغات يراعي اللغتين الإنجليزية والعربية. خلال السعي في تحقيق هذا الهدف ، جمعنا قدرا كبيرا من النصوص للغتين مختلفتين و قمنا بتهيئتها و تسويتها، ثم عملنا على إنشاء نواة فعالة للنظام تعرف باسم نموذج تضمين الكلمات. و استمر العمل نحو إنشاء نظام التشابه. في النهاية ، جمعنا بين النظام و ثلاث طرق ترجيح و وزن مختلفة محاولة منا لتحقيق مزيد من التحسين و الدقة.

الكلمات المفتاحية: معالجة اللغة الطبيعية ، تضمين الكلمات ، الترجمة الآلية ، التشابه الدلالي.

Contents

List of Figures

List of Tables

Introduction

In Natural Language Processing (NLP), Semantic Textual Similarity (STS) is one of the core tasks of many Computational Linguistics applications and their related areas. STS can be defined as a way to measure the degree of semantic and syntactic equivalence between two text units (texts, paragraphs or sentences). There are two known types of STS: monolingual and cross-lingual. The first predicts the semantic similarity based on two tokens written in the same language, how equivalent are they, while the cross-lingual aims to determine the degree of resemblance to which the two tokens are indicative, as well as independently of the languages these tokens are written in.

The similarity between sentences is determined fairly smoothly in a monolingual field. While it is some how difficult cross-lingually because the relationship between words is investigated through a change pace between two different languages. Therefore, improving more cross-lingual STS is required to thus improve performance in many real-world applications, such as Cross-Lingual Plagiarism Detection (CLPD) and Cross-Lingual Information Retrieval (CLIR).

In this work, we focus our investigation on measuring STS between Arabic-English sentences using word vector representations. We also consider words alignment and try to provide a suitable one, investigate weighting methods usage consequences on our system and try to choose a one that gives higher credibility to the built vector space. We seek developing a word integrating approach based on the measurement of the semantic similarity between Arabic-English sentences through exploiting the representation that assures capturing the syntactic and semantic properties of words while being in larger tokens like sentences.

The rest of this thesis is organized as follows. In Chapter 1, we present the State of the Art for STS and its supplements. In Chapter 2, we construct and examine a Cross-Lingual Word Embedding Model by performing some alignment and evaluation methods. In Chapter 3, we develop the Cross-Lingual Semantic Similarity system, with looking for improvements by applying some weighting aspects. Finally , the conclusion and future vision especially about alignment methods.

Chapter 1

State of the Art

"What's special about human language? It's the most important distinctive human characteristic, the only hope for explainable intelligence and it is a social system."

Christopher Manning

Introduction

Natural Language Processing is one of the earliest Artificial Intelligence fields, thus, its history has gotten wider and deeper. For someone who wants to propose a contribution in the field, it is highly required for him to keep up first. To do so, providing a State-of-art study is an ideal thought.

Empirical benchmarks in NLP field shows that considering words as the basic level token is a strong aspect to start with. Therefore, most of previous and current researches were word-level focus. In this chapter we studied the most famous word-based work known as Word Embeddings (WE) and their application in many NLP tasks.

This chapter is organized as follows. In Section 1.1 we define Word Embeddings, with describing its fields of use. We also present their different architectures. In Sections 1.2 we present the technical face of WE. Section 1.3 presents the linguistic dataset and its types. Section 1.4 presents already contributed Cross-Lingual Word Embeddings in the field.

1.1 Word Embeddings

1.1.1 Definition

Word Embeddings (WE), also known as *"Distributed Word Representation"* is a mathematical vector space that is supposed to guarantee displaying syntactic and semantic relationships of words, hence, the distance between them Levy and Goldberg (2014b). The d-dimensional vector of a word is decided due to this word's context. This same idea came up to light for the first time in the Distributional Hypothesis of Harris (1954), stating that : *"words in similar contexts have similar meanings"* Levy and Goldberg (2014a).

A common mistake in understanding Word Embeddings is that close words in the vector space are synonyms. Where actually, these words are just syntactically and/or semantically similar Mikolov et al. (2013a). Based on this, it is self-evident that distance between words is the keystone of recognizing sentences and documents similarities Levy et al. (2015).

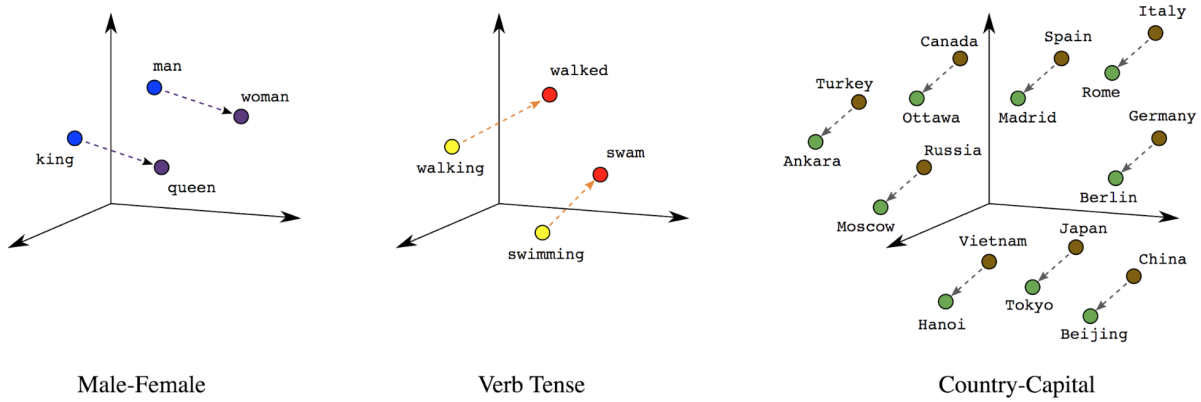

Figure 1.1: Relations between words according to word embeddings Mikolov et al. (2013a).

For the last two decades, word vector representation has been the " big bang " of Natural Language Processing world. Thus, improving it means moving this latter a further step Mikolov et al. (2013a). Many years experiments showed that focusing on collecting colossal amount of data for training vectorial models outplays the ones trained with sophisticated systems and less data Mikolov et al. (2013a).

1.1.2 Fields of Use

Natural Language Processing tasks often and widely consider treating words as a start line. But not the word as just a base level component, also relationships between words is a crucial point for them. That is exactly, what a Word Embedding Vector Space can provide Levy and Goldberg (2014b). Most of the last empirical researches were harnessed to benefit from the modern words representation and showed that NLP tasks performance was pretty convincing using it. We will walk through four of these tasks in our work:

1.1.2.1 Semantic Textual Similarity:

Semantic Similarity (STS) task is measuring how much a couple of textual tokens can be linguistically close. These tokens differ from words to sentences, even documents and maybe queries. Some empirical researches results acknowledged that STS systems with a word embedding model Core precede traditional ones with a step that cannot be underestimated Levy et al. (2015).

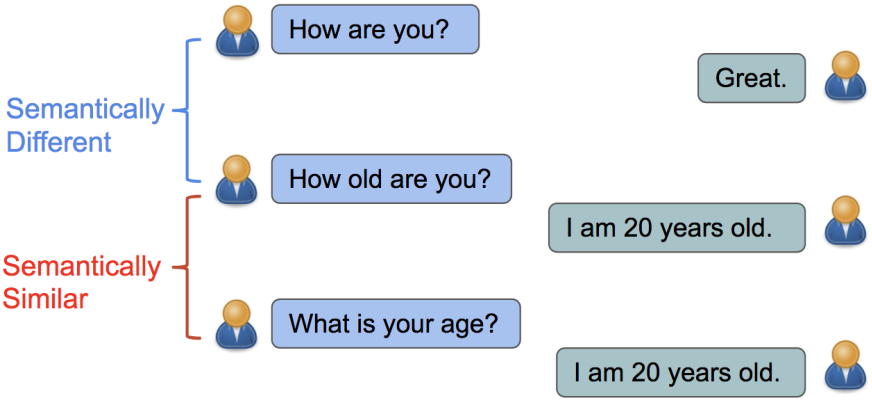

Figure 1.2: Using STS on chat conversation Yinfei Yang (2018).

1.1.2.2 Information Retrieval:

Referred to as IR, it is the process that seeks ranging, saving, serving and then easing the benefit from an information according to the search engine questioner's needs Bachchhav (2016). In *"Modern Information Retrieval"* book introduction, there is a convincing example of how much tricky this problem can be Baeza-Yates et al. (1999):

"Find all the page(document) containing information on college tennis teams which: (1) are maintained by an university in the USA and (2) participate in the NCAA tennis tournament. To be relevant, the page must include information on the natinal ranking of the team in the last three years and the email or phone number of the team coach Baeza-

Yates et al. (1999).”

Ivan Vulic and Marie-Francine Moens did a magnificent job with invloving Cross-lingual word embeddings in Information Retrieval Procedure, starting with preparing a specific Bilingual model which they named *”Bilingual Word Embeddings Skip-Gram”* oriented essentially for retrieving information sake, alongside multiple contributions that facilitate training model for Bilingual aspect. They asked a 3rd question: *”Are Word Embeddings useful in ad-hoc Information Retrieval?”*, and they answered indirectly with *”Yes”* by exploiting their word based model to build Query and Document Embeddings which are the heart of IR Vulić and Moens (2015).

1.1.2.3 Document Classification:

Document Classification has other several names like Text Categorization or Classification (TC) or even Text Spotting (TS). Its first appearance was because of two main events at the same time period: Information Retrieval task became highly important. The emerge of new robust hardware and the invention of neural nets and Machine Learning Sebastiani (2002).
Taking an example where WE and TC have crossed roads, Word Mover's Distance (WMD) exhaustively exploited the vectors in WE trained model to calculate how far a document words should travel to be equal to a second document words. That means the less words travel to be equivalent to other words in a different document, the closer documents are. Due to that, the achievement was a new lowest error rates in K-Nearest Neighbor Algorithm classifying documents compared to TC systems constructed before 2015 Kusner et al. (2015).

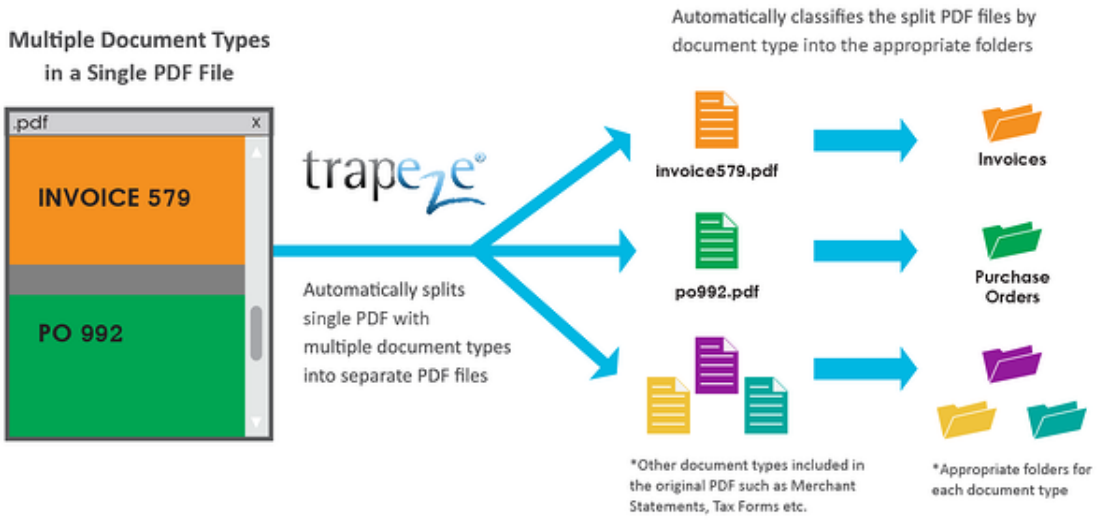

Figure 1.3: Extracting multiple documents from single PDF file [1].

1.1.2.4 Plagiarism Detection:

Plagiarism became a global problem that deserves the absolute priority of solving since the birth of Internet. This Latter is destined to be the number one source of all kind of information in any field. Due to that, In research field any one can re-use the hard work of others willingly without citing or mentioning them. In the present, Monolingual

plagiarism detection has a rich contributions list, such as the novel Plagiarism Detection System for Arabic Khorsi et al. (2018) composed of many critical detection levels, comparing to Cross-Lingual, which is harder to capture considering that the detected information is no longer in its source language. However, Jeremy Ferrero's team have developed a powerful cross-lingual Plagiarism Detection System using Sentences Semantic Similarity based on Word Embeddings Ferrero et al. (2017).

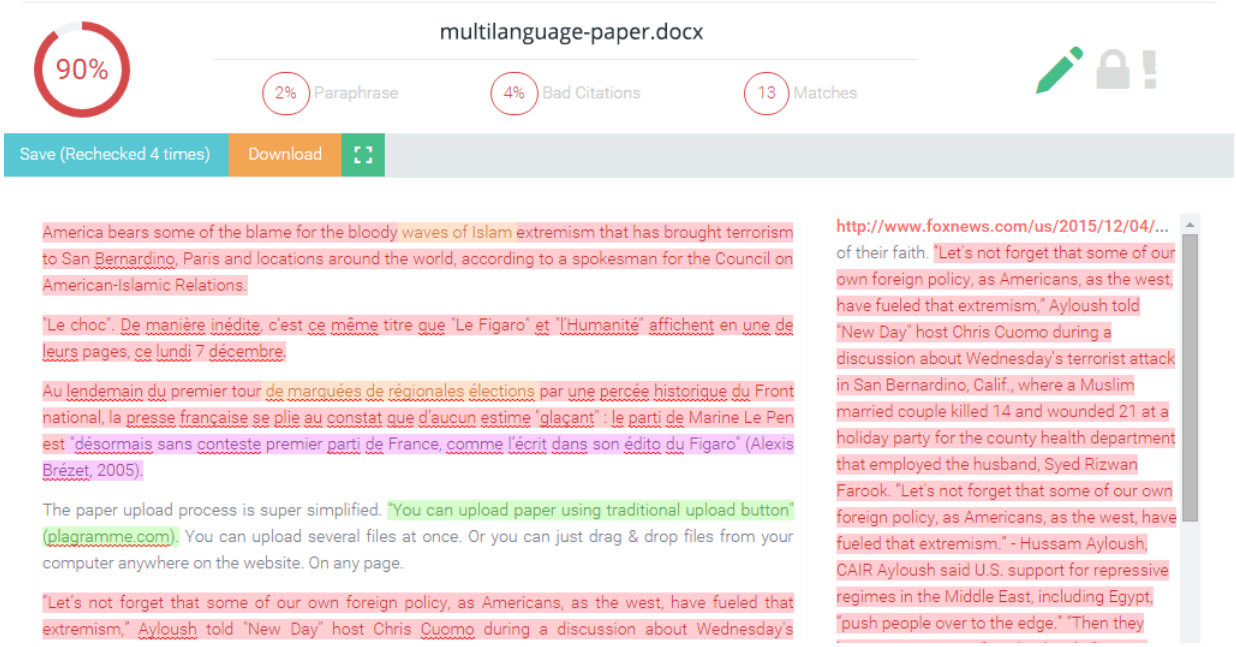

Figure 1.4: Screenshot for an online plagiarism detection tool [2].

1.1.3 Different Architectures

Word Embeddings have had a big share of researchers and developers time. That served a lot of different architectures which differ in terms of advantages and negatives. They are also distinguishable considering the way of choosing a word's context, this latter size and even how far it should be from the pivot word.

1.1.3.1 Collobert and Weston (2008)

Collobert and Weston presented an "All in One" system based on Deep Neural Net architecture. They trained Part of Speech Taggers, Chunkers, Semantic Role Labelers with labeled datasets. Our concern is the last, which is training a word embedding model considering unsupervised learning, that means using unlabeled data. We believe their attempt was the first of its kind, as they referred to in their paper. The reason behind choosing unlabeled data over the labeled was trying to skip the expensive, under preprocess hat, labeling stage that requires linguistic experts Collobert and Weston (2008).

1.1.3.2 Mnih and Hinton (2009)

This work frankly criticized the widely used Neural Probabilistic Language Models (NPLMs) Morin and Bengio (2005) in the previous decade due to the expensive training time cost and that it resulted a less accurate model compared to its non-hierarchical ancestor. After that, the authors presented their contribution by replacing the n-grams rigid representation with a more meaningful one based on real-valued vectors that should be similar only for words with similar contexts Mnih and Hinton (2009).

1.1.3.3 Turian et al. (2010)

Turian work made it very clear that involving unsupervised learned Word Embeddings in NLP systems improves their accuracy, and that this latter is directly proportional to the WE model's quality. They followed this previous analysis with a contribution informs that combining word representations can make NLP systems more reliable in varying proportions Turian et al. (2010).

1.1.3.4 Mikolov et al. (2013)

Mikolov's idea implementation gave birth to a revolutionary Word Embedding model. Two new opposite, but with same aspect, based on Neural Nets, architectures reduced time cost and served a larger efficiency for Word Similarity task Mikolov et al. (2013a). The most important is what happened after this innovation, systems learning architectures became less interesting and experiments confirmed that the biggest load is located in choosing an appropriate Corpora and pre-process it as needed to guarantee a well performing WE model Ruder et al. (2017).

1.1.3.5 Pennington et al. (2014)

This architecture is known by Global Vectors (GloVe). Researchers who own that contribution tried to include little of both from the two major model concepts: Global Matrix Factorization that deals usually with words co-occurrence across the whole treated corpus. The second is Shallow Local context window that represents words according to their surroundings. They clarified disadvantages of each method, represented their combined system then calculated its complexity and compared it with the state-of-art models in two NLP tasks: Semantic Similarity and Named Entity Recognition Pennington et al. (2014).

1.2 Word2Vec

Word2vec is a brainchild of a team of researchers led by Google's Tomas Mikolov. It is one of the most popular algorithms used to create word embeddings. This note provides detailed derivations and explanations of the parameter update equations of the word2vec algorithm, including the original continuous bag-of-word (CBOW) and skip-gram (SG) architectures, as well as advanced optimization techniques, including Hierarchical Softmax and Negative Sampling Rong (2014).

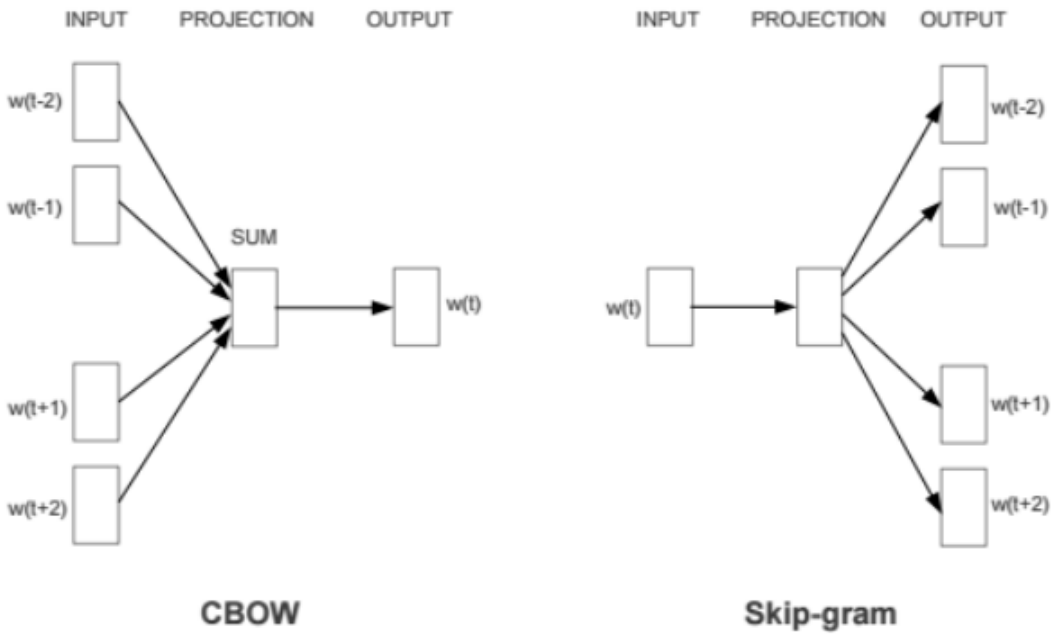

Figure 1.5: Description of the CBOW and Skip-gram models as described in Mikolov et al. (2013b).

1.2.1 Continuous Bag Of Words (CBOW)

CBOW contains two versions, the first being the simplest version presented in Mikolov et al., Which is only one word in context, while the second version, a multi-word context.

- **One-Word Context**

Starting with a simple CBOW style where textual features are captured from one context word, usually the one before the pivot word, as illustrated in Figure 1.6. In general, in context, the vocabulary size is V and the size of the hidden layer is N Rong (2014).

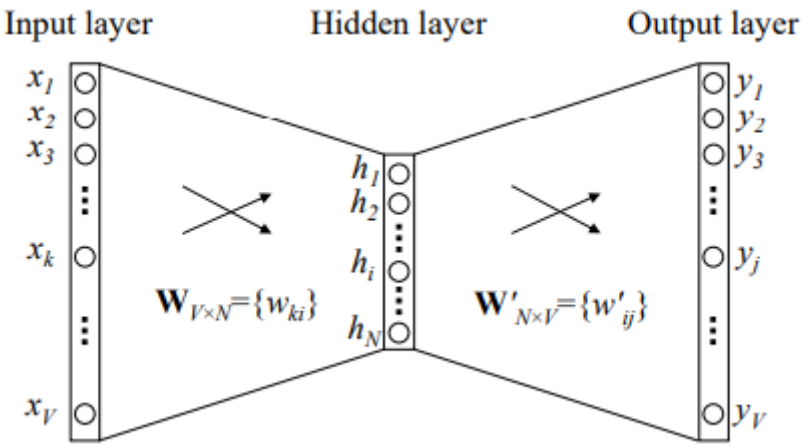

Figure 1.6: A simple CBOW model with only one word in the context Rong (2014).

"The units on adjacent layers are fully connected. The input is a one-hot encoded vector, which means for a given input context word, only one out of V units $\{x_1, \cdots, x_V\}$ *will be* 1*, and all other units are* 0*."* Rong (2014)

- **Multi-Word Context**

The figure 1.7 shows the CBOW model when dealing with multiple context words, so instead of feeding each word one-hot vector separately it is required to calculate their average vector first. Rong (2014).

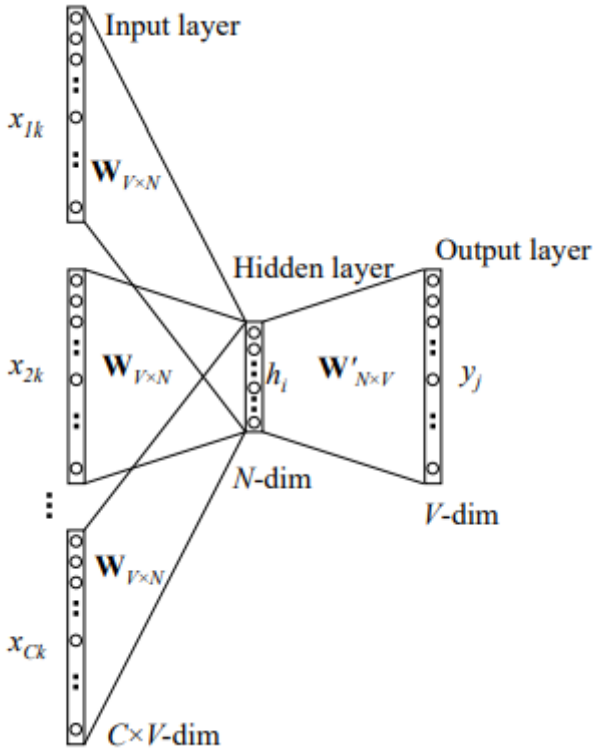

Figure 1.7: Continuous bag-of-word model with Multi-Word in the context Rong (2014).

"... where C is the number of context words, $w_1, \cdots, w_C$ *are the context words and* v_w *the input vector of a word w."* Rong (2014)

1.2.2 Skip-gram Model

As Skip-Gram is the reverse of CBOW, it captures the features provided by one pivot word trying to predict one or multiple context words, this aspect gave better results some prediction procedures like *next token prediction*, *previous token prediction* and *SQUAD relationships capturing* Rong (2014).

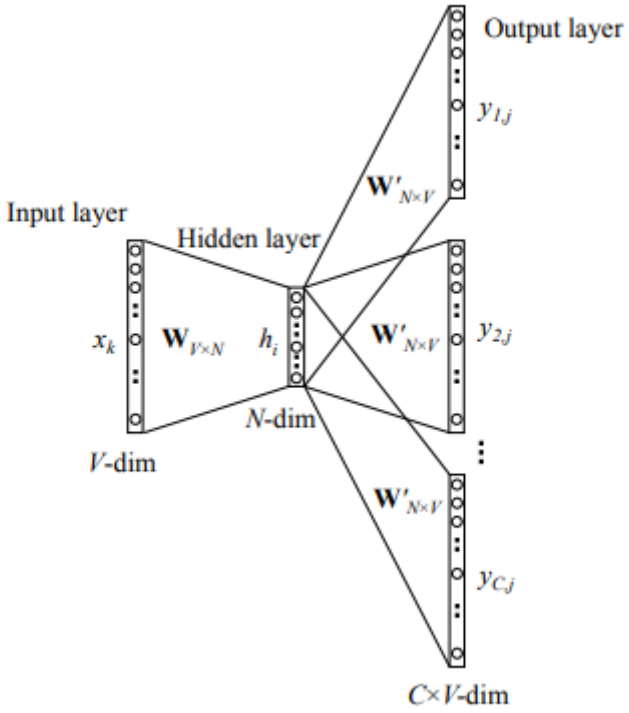

Figure 1.8: The skip-gram model Rong (2014).

1.2.3 Word2Vec Parameters

1.2.3.1 GenSim

GenSim is an open source project that provides an efficient tool for training a large text collection in an unsupervised manner and with a lot of flexibility. For instance, for a researcher that does not acquire a hardware with machine learning recommended features can separate his training on many phases.Řehřek and Sojka (2011).

```
3 from gensim.models import Word2Vec
```

Figure 1.9: Word2Vec class in GenSim library

1.2.3.2 Parameters

Word embedding models require some parameters that affect the resulting vector. Thus, the CBOW model we used the following parameters, which are used by Mikolov et al. Mikolov et al. (2013b).

- sentences (iterable of iterables, optional)= The sentences iterable can be simply a list of lists of tokens, but for larger corpora, consider an iterable that streams the sentences directly from disk/network.

- min count (int): Ignores all words with total absolute frequency lower than this (2, 100).

- window (int): The maximum distance between the current and predicted word within a sentence. E.g. window words on the left and window words on the left of our target (2, 10).
- size (int): Dimensionality of the feature vectors (50, 300).
- negative (int): If > 0, negative sampling will be used, the int for negative specifies how many "noise words" should be drown. If set to 0, no negative sampling is used;(5, 20).
- workers (int): Use these many worker threads to train the model (faster training with multicore machines).
- sg (0, 1, optional): Training algorithm: 1 for skip-gram; otherwise CBOW.
- total examples (int): Count of sentences.
- epochs (int): Number of iterations (epochs) over the corpus.

1.3 Corpora

The used Corpora, which is the linguistic dataset, differs from work to another. Precisely, the method of alignment and the way of getting the translation of a specific language data. Starting with the two general concepts, Parallel or Comparable. Both of them have the same three branches as the following:

1.3.1 Parallel Corpora :

Parallel corpora is the group formed by text and translated. Parallel corpora alignment is the task of specifying correspondence between blocks or tokens in each half of the bit text. Alignment is used in several different areas of Linguistic and Computational Linguistic Research, and has several categories.

1.3.1.1 Word Alignment:

The more common and used so often. Usually available in a form of cross-lingual words dictionary Ruder et al. (2017) like:

{'cat': 'قطة', 'car': 'سيارة', 'bus': 'حافلة','knife': 'سكين','does': 'يفعل','warm': 'دافئ'}.

Note: this type of alignment is present in Comparable Corporas also. But, we will not mention it since they do not differ from each other.

1.3.1.2 Sentence Alignment:

This type has got its attention coinciding by the first appearance of Mikolov's Word2Vec approach. It consists of the first language sentences and their exact translation in the

second one Ruder et al. (2017). it is not necessarily a manual (human) translation, it could be a machine translation as long as it is relatively accurate.

```
    <tu>
      <tuv xml:lang="ar"><seg>في الأسلحة النووية، في الجلسة العامة لمؤتمر الأطراف
2005 أيار/مايو 11 نيويورك، بشأن إقرار جدول الأعمال، 2005 عام المعاهدة لاستعراض
      <tuv xml:lang="en"><seg>Statement by the delegation of Malaysia, on behalf
of Nuclear Weapons, at the plenary of the 2005 Review Conference of the Parties
of the agenda, New York, 11 May 2005</seg></tuv>
    </tu>
```

Figure 1.10: Sentence-aligned parallel corpus Rafalovitch et al. (2009).

1.3.1.3 Document Alignment:

People in the field usually do not encounter this alignment because it literally means documents are the exact translation of each other and intuitively achieving that requires being these documents, at least, sentence-aligned Ruder et al. (2017).

1.3.2 Comparable Corpora:

A Comparable Corpus is a collection of similar texts in different languages or in different varieties of a language. Aligning comparable corpora automatically would provide a valuable resource for learning of text-to text rewriting rules.

1.3.2.1 Sentence Alignment:

The difference between Comparable Sentence-Aligned and Parallel Sentence-Aligned is that the former contains non-exact translation. In other words, translation in opposite sentences does not go word by word. it is sufficient that both of them have a common usage or touch the same subject and/or object Ruder et al. (2017).

1.3.2.2 Document Alignment:

It is worth it to mention that the most famous resource for this type is the Wikipedia Encyclopedia. For example, Aligning documentation of the topic "Emotional Intelligence" ,for both English and Arabic, will make it pretty clear that the english instance is a lot larger and richer considering the number of words. But that does not contradict with using it as a Comparable Doc-Aligned corpus Ruder et al. (2017).


ذكاء عاطفي

الذكاء العاطفي هو القدرة على فرز العواطف الذاتية، وحسن استعمالها. ويمكن تقسيمها إلى القدرة *EI* و *التجاوز* EI.
تركزت الانتقادات على ما إذا كان EI هو حقيقي الذكاء وما إذا كان لديه سريان تراكمي نسبة الذكاء و سمات الشخصية
الخمس الكبرى[1] ويعرف **كولمان Goleman** الذكاء العاطفي بأنه القدرة على التعرف على شعورنا الشخصي وشعور
الآخرين ، وذلك لتحفيز أنفسنا، ولإدارة عاطفتنا بشكل سليم في علاقتنا مع الآخرين.

Emotional intelligence

From Wikipedia, the free encyclopedia

Emotional intelligence (**EI**), **emotional leadership** (**EL**), **emotional quotient** (**EQ**) and **emotional intelligence quotient** (**EIQ**), is the capability of individuals to recognize their own emotions and those of others, discern between different feelings and label them appropriately, use emotional information to guide thinking and behavior, and manage and/or adjust emotions to adapt to environments or achieve one's goal(s).[1][2]

Figure 1.11: Document-aligned comparable corpus.

(a) Word, par. (b) Word, comp. (c) Sentence, par. (d) Sentence, comp. (e) Doc., comp.

Figure 1.12: Examples for the nature and type of alignment of data sources. Par.: parallel. Comp.: comparable. Doc.: document. From left to right, word-level parallel alignment in the form of a bilingual lexicon (a), word-level comparable alignment using images obtained with Google search queries (b), sentence-level parallel alignment with translations (c), sentence-level comparable alignment using translations of several image captions (d), and document-level comparable alignment using similar documents (e) Ruder et al. (2017).

1.3.3 Seed Lexicons and their importance:

Considering the difference between parallel and comparable alignments, that should not be all for the latter one. A crucial phase in any cross-lingual NLP task is locating Named Entities (proper names, locations, organizations...etc). Doing this in a language pair comparable corpus seems more challenging especially in the case one of the languages is a low-resource. For the purpose of facilitating, a Bilingual Seed Lexicon serves as a bridge between the pair. It usually shows up as an entities dictionaryWang et al. (2013).

1.4 Cross-Lingual Word Embedding Models

There are multiple Bilingual Word Embeddings concerning different Language pairs and trained by different WE architectures. In this thesis, we will go only through the Crosslingual WE models based on Miklov's Architectures (Skip-Gram, Continuous Bag of Words):

1.4.1 Bi-Lingual Word Embedding Skip-Gram (BWESG)

This model is constructed through three main steps:

1. **Document Skip-Gram:** This step requires constructing a Skip-Gram architecture that deals with documents as inputs Vulić and Moens (2015) after pre-processing them as explained in the next step.

2. **Corpus Preprocessing and Shuffling:** This work's comparable corpus was a Wikipedia article pairs collection of the same topic each and for different languages (Spanish, Italian, German and English). This corpus has been normalized, pre-processed, cleaned from sentences boundaries and then each bilingual pair was randomly shuffled to have a so called *"Pseudo-Bilingual Document"* that serves as one of the inputs Vulić and Moens (2015).
   Note: the corpus was shuffled randomly for 10 different times to assure shuffle significance in every one and that it was not "a stroke of luck"Vulić and Moens (2015).

3. **Train BWESG Model** In the training phase, Word2Vec Skip-Gram parameters Vulić and Moens (2015) were considered:

   - fix learning rate to 0.025.
   - 25 for the Negative Sampling.
   - $1e-4$ for subsampling.
   - 15 training epochs.
   - Vector dimension had multiple values, to realize experiments aspect, which are $d = 100, 200, 300$ and 40.
   - Context window size also had multiple variants in 4 to 60 words range.

1.4.2 Bilingual Skip-Gram (BiSkip)

The BiSkip model was trained on a German-English sentence aligned parallel corpus. Such a choice have made it relatively simpler to preserve the model's monolingual quality while focusing on the bigger goal: construct a robust Bilingual WE ModelLuong et al. (2015).
Leaving the math away for now, training the corpus has depended on the concept of reversing both the input and objective of languages while training:

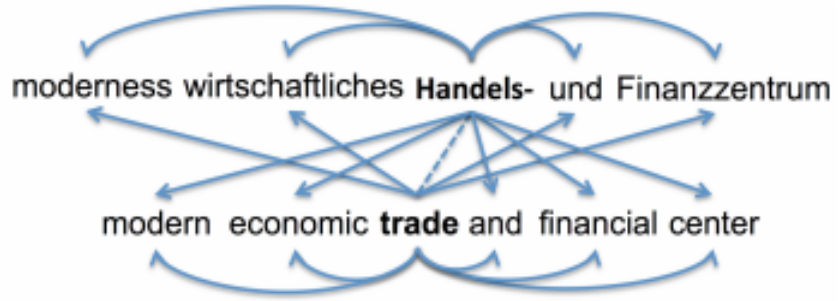

Figure 1.13: An example describes BiSkip switching where "Handels" is the translation of "trade"Luong et al. (2015).

This switching has made four combinations to be trained and the results were summed up in some kind of Mono Bi-Lingual Model that assures monolingual tasks reliability despite the small corpora comparing to the original Mikolov's word2vec, plus coming up with an outperforming Cross-Lingual modelLuong et al. (2015).

1.4.3 ADAN English-Arabic model

Adversarial Deep Averaging Networks (ADAN) is a sentiment classification project that raised the problem of unlabeled low-resource target language data and and proposed a solution for it. In their experiments phase they chose English as a SOURCE language, both Chinese and Arabic as TARGET language for building their system's WE model core Chen et al. (2018).

In training their English-Arabic model, they used UN corpus Ziemski et al. (2016) and trained it with BilBOWA WE architecture Gouws et al. (2015).

Bilingual Bag-of-Words without Alignments (BilBOWA) is an architecture that aims to train two monolingual WE models with at the same time minimizing the distance between translation pairs vector representations by aligning their words Gouws et al. (2015).

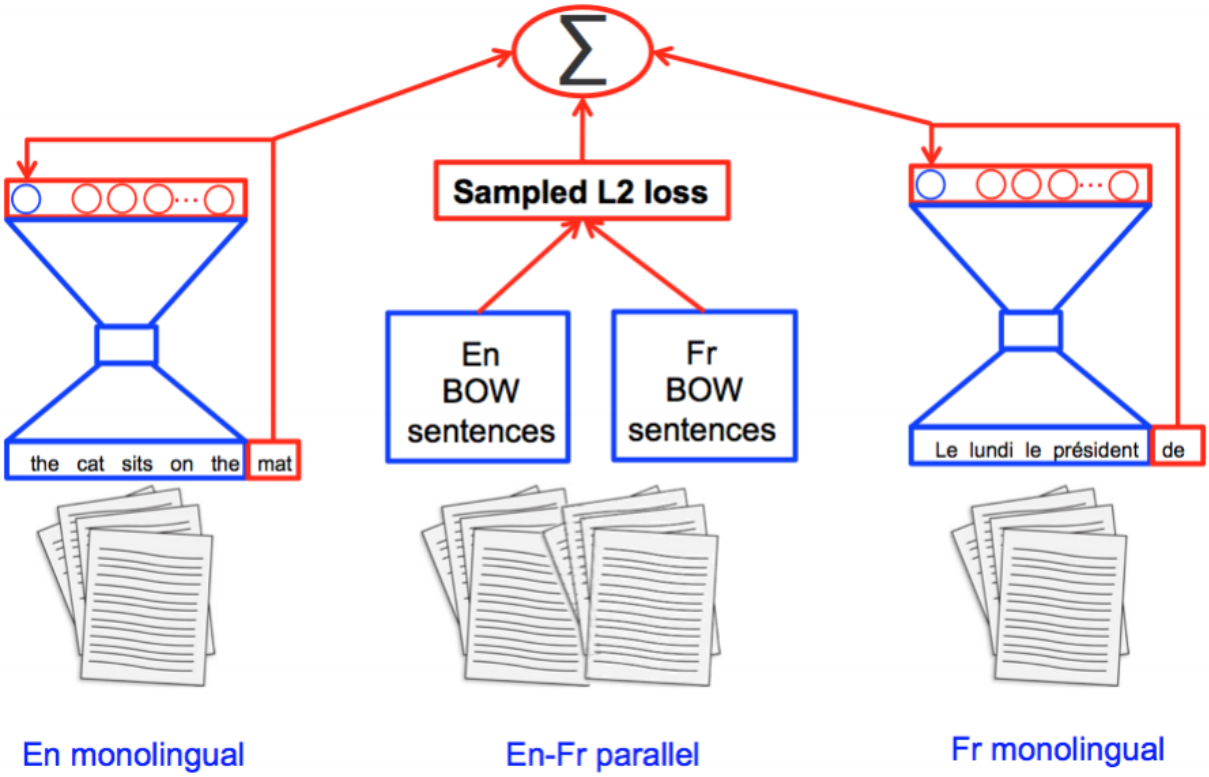

Figure 1.14: Schematic of the proposed BilBOWA model architecture for inducing bilingual word embeddings Gouws et al. (2015).

1.4.4 Multilingual BERT

Bidirectional Encoder Representations from Transformers (BERT) is NLP state-of-the-art architecture. It was mainly produced to overcome the limitation of word embeddings such as restricting prediction into either *next word prediction* or *previous word prediction*, starting from masking some of the input tokens which is a sentence pair Devlin et al. (2018). The first step of BERT is to tag the start of the first sentence in the trained pair with the token $[CLS]$ and its end by $[SEP]$ and the second sentence tagged only in the end also by $[SEP]$. Then 15% of the tokens are masked with $[MASK]$. the architecture aims to predict these masked tokens based on its left and right surroundings.

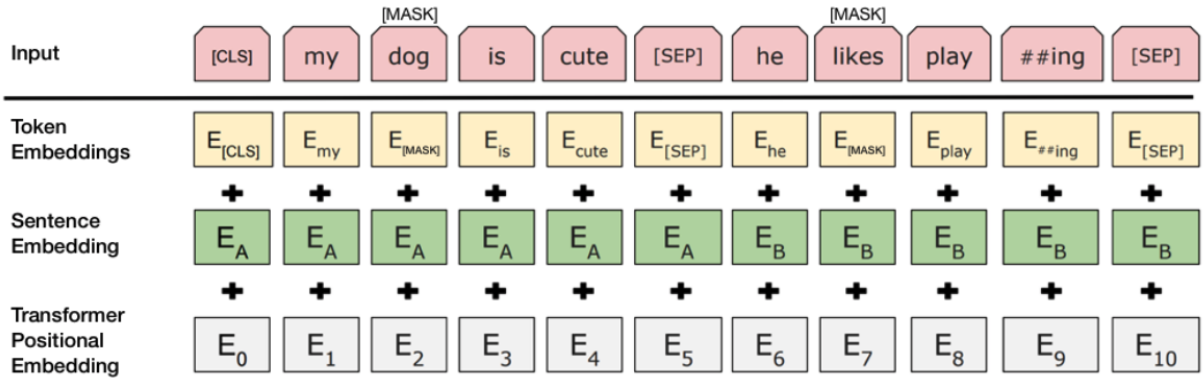

Figure 1.15: BERT tokens prediction (source image modified by Rani Horev) Devlin et al. (2018).

BERT also aims to predict next sentences. More specifically, predicting if the second sentence in the fed sentence pair is actually the original subsequent sentence for the first in the corpus. This is done by feeding 50% of the pairs as real sequences and the rest by randomly choosing the second sentence of the pair and then calculate the appearance probability with $softmax$. Devlin's team extended their work by implementing BERT on a multilingual wikipedia corpus that covers more than 104 different languages (mBERT) and generously make it as an open source project.

Conclusion

In this chapter, we defined the concept of Word Embeddings, some NLP tasks that are based on it and their performance, As well as a brief description of different architectures of WE, the ones they have had a big share of researchers and developers time. We also presented $Word2Vec$, its methods(CBOW, Skip-Gram) and parameters. Furthermore, we described the types of corpora and role of alignment. Finally we presented the most important works of cross-lingual WE models which are based on Mikolov's Skip-Gram, Continuous Bag of Words and other state-of-the-art contributions.

Chapter 2

Cross-Lingual Word Embedding Model

"Chat Bots represent the next evolution in NLP and they provide a cheap and effective way to answer simple questions and provide a 24/7 access point between businesses and customers. Time to embrace our robot overlords."

Casey Markee

Introduction

The first thing the researcher could benefit from providing a state-of-art survey is widening his options. Lately, many NLP tasks have been enriched using tools based on Cross-Lingual Word Embeddings (CLWE). CLWE enable us to reason word meaning in multilingual contexts and is a key facilitator of cross-lingual transfer when developing linguistic systems especially for low-resource languages.

Many CLWEs have been developed, particularly for English as the best source language, down to many low resource targets, but Arabic did not get that much of interest. In this chapter, to train our model we extracted more than 93.9 million parallel Arabic-English sentences from whole *OPUS* collection, this alignment contains more than 800 million Arabic tokens with 1 billion for English.
The following is how this chapter is organized: in Section 2.1 we provide a quick overview of our chosen architecture.We describe our dataset collection and the preprocessing procedure in Section 2.2. Section 2.3 presents the essential environment we used while developing our model. Section 2.4 presents the steps of preprocessing our corpus. Section 2.5 contains the ways and steps for training our model variants. Finally Section 2.6 presents how the model variants were evaluated and the results.

2.1 Chosen Architecture

2.1.1 Word2Vec

Word2Vec is the most common nickname for Mikolov's Skip-Gram and CBOW architectures. It actually first came up with the research team's software appearance Goldberg and Levy (2014).

2.1.2 Why Word2Vec ?

Empirical results made it clear that Mikolov's models are on top of accuracy rank mono and cross-lingually especially in Semantic Similarity task which our thesis clearly corresponds to. More than that, a lot of contributions, that we could benefit from, have been added the last years used this same architecture Ruder et al. (2017). It is worth mentioning, that a specific paper was so useful in constructing our similarity system and saved us a lot of time catching up. They used the "Arabetized Mikolov" Zahran's model Zahran et al. (2015) as their system's core, and they included important weighting improvements alongside mere system results Schwab et al. (2017).

2.2 Collection of Corpora

To train our model, we built our own corpus using a large collection from different sources extracted from the Open Parallel Corpus Project (OPUS) [1]. we extract more than 93.9 million parallel sentences of Arabic-English from whole collection, this alignment contains more than 800 million Arabic tokens alongside 1 billion for English.

[1] http://opus.nlpl.eu/

2.2.1 OPUS

OPUS Tiedemann (2012) is a large collection of translated texts from the web. OPUS is an open web resource that tries to convert and align free data over the Internet, to add language commentary, and to provide the community with a parallel presentation available to the public.It contains 90 languages, and more than 2.7 billion parallel sentences, this corpus consists of data from multiple domains and sources including : MultiUN Corpus Eisele and Chen (2010), OpenSubtitles Creutz (2018), Tanzil ZarrabiZadeh (2007), News-Commentary, United Nations (UN) Ziemski et al. (2016),TED 2013 [2], GNOME [3], Tatoeba [4],Global Voices [5],KDE4 [6] and Ubuntu [7] corpus.

- **Wikipidia**

  This is a corpus of parallel sentences taken from Wikipedia by Krzysztof WoÅk and Krzysztof Marasek.It includes a total of 36 bitexts extracted from a collection of 25 million sentences (610 million tokens) distributed over 20 languages Wołk and Marasek (2014), we used $0.2M$ sentences parallel from arabic english corpus, containing $3.2M$ arabic words and $3.5M$ english words.

- **MultiUNCorpus** This is a collection of translated documents from the official documents of the United Nations (UN), by Andreas Eisele and Yu Chen.This corpus is available in all 6 official languages of the UN, consisting of around 300 million words per language Eisele and Chen (2010),we used 67617 documents from arabic english corpus, containing $10.6M$ sentences parallel, including $263.1M$ arabic words and $289.6M$ english words.

- **OpenSubtitles** This is a new collection of translated movie subtitles Creutz (2018), we utilized 104325 documents from arabic english corpus, consisting $81.4M$ sentences parallel, including $501.5M$ arabic words and $695.9M$ english words.

- **Tanzil** This is a collection of Quran translations compiled by the Tanzil project[8], we used 30 documents (Quran Party), consisting $0.2M$ sentences parallel, including $7.9M$ arabic words and $5.6M$ english words.

- **News-Commentary** This is a parallel corpus of News Commentaries provided by WMT for training Statistical Machine Translation (SMT), we used 7185 documents, consisting $0.6M$ sentences parallel, including $15.4M$ arabic words and $15.5M$ english words.

- **UN** This is a collection of translated documents from the United Nations originally, containing $74.1k$ sentences parallel, including $3.3M$ arabic words and $3.7M$ english words.

[2] http://www.casmacat.eu/corpus/ted2013.html
[3] https://l10n.gnome.org
[4] www.tatoeba.org
[5] https://globalvoices.org/
[6] http://i18n.kde.org
[7] https://translations.launchpad.net
[8] http://tanzil.net

- **TED2013** This is a parallel corpus of TED talk subtitles provided by CASMACAT[9], containing $0.2M$ sentences parallel, including $2.4M$ arabic words and $3.0M$ english words.
- **GNOME** This is a parallel corpus of GNOME localization files, consisting 1313 documents and 0.5M sentences parallel, including $2.5M$ arabic words and $2.6M$ english words.
- **Tatoeba** This is a collection of translated sentences from Tatoeba[10], consisting $13.0k$ sentences parallel, including $90.1K$ arabic words and $3.6M$ english words.
- **GlobalVoices** This is a parallel corpus of news stories from the web site Global Voices[11], consisting 7017 documents and $93.9k$ sentences parallel, including $2.1M$ arabic words and $3.0M$ english words.
- **KDE4** This is a parallel corpus of KDE4 system messages, consisting 784 documents and $0.1M$ sentences parallel, including $0.7M$ arabic words and $0.8M$ english words.
- **Ubuntu** is a parallel corpus of the Ubuntu Dialogue Corpus, containing 299 documents and $56.3M$ sentences parallel, including $0.2M$ arabic words and $0.5M$ english words.
- **EUbookshop** This is a Corpus of documents from the EU bookshop,an online service and archive of publications from various European institutions,it contains 30 documents and $1.7k$ sentences parallel, including $80.0K$ arabic words and $0.4M$ english words.

we used a parallel corpus ,it's a corpus that contains a collection of original texts of language L_1 and their translations into a set of $L_2...L_n$ languages.In order to use a parallel corpus properly it is necessary to align the source text and its translation(s). This means that one has to identify the pairs or sets of sentences, phrases and words in the original text and their correspondences in the other languages.
Parallel text alignment is important because during the translation process sentences might be split, merged, deleted, inserted or reordered by the translator in order to create a natural translation in the target language. In order to compare the original text and its translation(s), In the process of alignment, anchor points such as proper names, numbers, quotation marks etc. are often used as a points of orientation.
Text has been matched as far as possible in terms of type of text, subject, and good translation, so it can be used as a parallel corpus or as a comparable corpus, we loaded and working for it on several styles as plain text or tmx, tgz, xml ..., and some corpus in the text format was Convert it into xml format, and compile it in the form of a single corpus consisting of about one billion words and of the most important areas of the corpus used: politics, economics, sociology, and culture.
We propose a multipurpose preprocessing framework that can be used to systematically transform our raw ingested text into a form that is ready for computation and modeling. Our framework includes the five key stages shown in figure 2.1 : pairs extraction, sentence segmentation, omitting tags, normalizing arabic, alignment. For each of these stages.

[9] http://www.casmacat.eu/corpus/ted2013.html
[10] http://tatoeba.org/
[11] https://globalvoices.org/

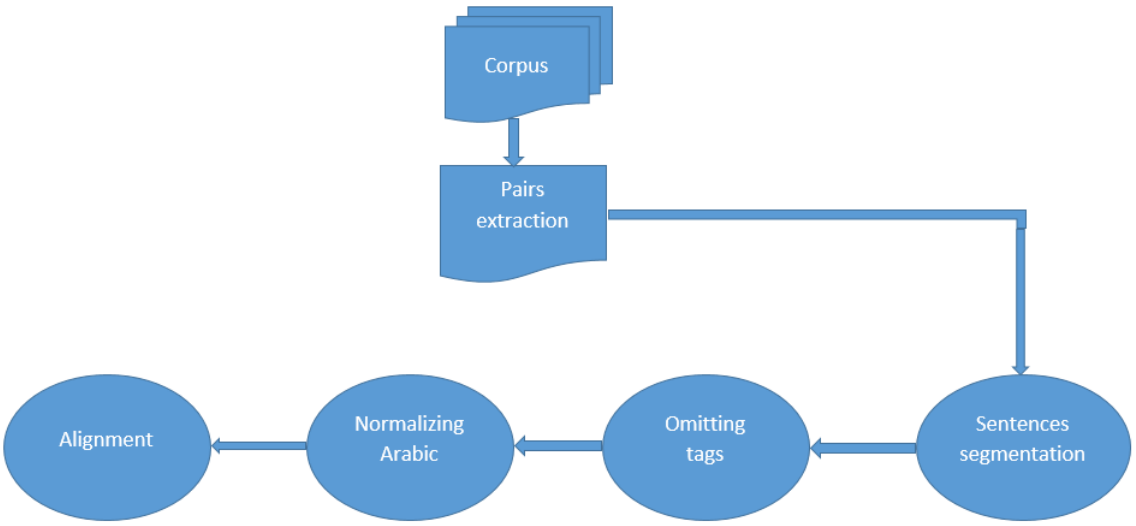

Figure 2.1: The key stages of preprocessing framework.

More generally, we are interested in taking predetermined body of text and performing upon it some basic analysis and transformations, in order to be left with artefacts which will be much more useful for performing some further, more meaningful analytic task afterward. This further task would be our core text mining or natural language processing work. So, there are two main components of text preprocessing: tokenization, normalization , as we layout a framework for approaching preprocessing.

2.3 The Environment of Choice

Choosing and preparing an appropriate developing environment can certainly save researchers time. A satisfying environment can assure a comfortable researcher, hence, continuity and more working time.

2.3.1 Google Colaboratory

Google Colaboratory Cloud Service, widely known as *Colab*. It is based on *Jupyter Notebook* as its environment. It is a perfectly prepared development environment that does not need a thing installed but a browser. And most importantly, almost $12GB$ of GPU completely free. One can also mount his personal Google Drive so he becomes able to import and/or save files or even deal with it as the hard drive of his local machine Carneiro et al. (2018).

2.3.2 Use Google's Internet

fact, *Colab* is also Linux based environment. In other words, one can use Linux commands easily as part of the code. For instance, downloading large datasets to google drive in no time using $!wget$ command with a very fast internet debit above $40MB/s$.

2.4 Corpus Preprocessing

Preprocessing the corpus differs depending on many facts, like the used language , or languages, in constructing the corpus because for instance, normalizing an English text tasks cannot be sufficient in normalizing an Arabic one. Mentioning an other restriction to make it clear, dealing with an XML tagged and aligned file is different comparing to JSON files. figure 2.2 is the code we used for preprocessing our corpus.

```
def arabic_preprocesser(line):
  line = stopWordsRemover(line)
  # remove commas and points
  nLine = ""
  for char in line:
    if char not in [u'.', u'،']:
      nLine += char
  line = nLine
  # remove_diacritics
  regex = re.compile(r'[\u064B\u064C\u064D\u064E\u064F\u0650\u0651\u0652]')
  line = re.sub(regex, '', line)

  # remove_urls
  regex = re.compile(r"(http|https|ftp)://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+")
  line = re.sub(regex, ' ', line)
  # remove elongation
  regex = re.compile(r'\u0640')
  line = regex.sub('', line)
  # remove_numbers
  regex = re.compile(r"(\d|[\u0660\u0661\u0662\u0663\u0664\u0665\u0666\u0667\u0668\u0669])+")
  line = re.sub(regex, ' ', line)

  # noramlize
  regex = re.compile(r'[|ٱ|إ|أ]')
  line = re.sub(regex, 'ا', line)
  regex = re.compile(r'ا+')
  line = re.sub(regex, 'ا', line)
  regex = re.compile(r'[ى]')
  line = re.sub(regex, 'ي', line)
  regex = re.compile(r'[ئ]')
  line = re.sub(regex, 'ء', line)
  regex = re.compile(r'[ؤ]')
  line = re.sub(regex, 'و', line)
  regex = re.compile(r'[ة]')
  line = re.sub(regex, 'ه', line)
  # remove one_character words
  regex = re.compile(r'\s.\s')
  line = re.sub(regex, ' ', line)
  line = ' '.join([word for word in line.split() if not re.findall(r'[^\s\u0621\u0622\u0623\u0624\u0625\u0626\u

  return line
```

Figure 2.2: Arabic Preprocessing code.

Preprocessing is normalizing the content of the data into a unified characters use. Ours has included multiple normalizition tips which they were inspired from Schwab et al. (2017). Such as removing elongations, for example مـــاليـــزيـــا would look like ماليزيا in arabic sentences, lowercasing for english ones, and globally deleting digits.

After preprocessing and normalizing Arabic, we applied GenSim's *simple_preprocess* function on both languages to get a ready-for-training input version.

2.4.1 TMX Extension Choice

2.4.1.1 What is ".tmx" anyway ?

Translation Memory eXchange (TMX) is spread from XML, exploiting its flexibility, security, and ease of transmission along the Web. It first appeared in 1998, and used to preserve translation memories like Multilingual Corpora Joshi and Mathur (2012).

2.4.1.2 Why tmx files?

Websites that provide corpora collections serve multiple choices concerning extensions of files holding the corpus. To us, choosing TMX files was more convenient due to the advantage of using XML tags in extracting sentence pairs.

2.4.2 Extracting Sentence Pairs

```
# deal with sentence pairs
def sentencePairReader(partition):
  regex = re.compile(r'<tuv xml:lang="ar"><seg>.*[\u0600-\u06FF]+.*</seg></tuv>')
  x = regex.findall(partition)
  regex = re.compile(r'<tuv xml:lang="en"><seg>.*\s*.*</seg></tuv>')
  y = regex.findall(partition)
  regex = re.compile(r'(<tuv xml:lang="ar"><seg>|<tuv xml:lang="en"><seg>|</seg></tuv>)')
  if x != []:
    x = regex.sub('', x[0])
  else:
    x = ""
  if x != []:
    y = regex.sub('', y[0])
  else:
    x = ""

  xy = arabic_preprocesser(x) + "\t" + y

  return xy
```

Figure 2.3: Receiving corpus chunk and getting rid of tags

2.4.2.1 Useless XML reader !

As we mentioned before, our XML-based files choice was to benefit from tags parsing and manipulating. We have chosen Python's Element Tree XML API to do so. But, since data size was huge, reading the corpus all at once wasn't an option and led our *Colab* environment to crash.

2.4.2.2 Switching to Regular Expressions

The available solutions to read our corpus were limited, either find other XML reading format package that does not require uploading the whole file to RAM, which is, to our knowledge, not available or separate the file into smaller files but that would cause us loosing *tmx* file structure. So we improvised using regular expressions for identifying sentence pairs starting from their form in *tmx* format as illustrated in previous figure 1.10.

```
# regexG pattern is used for identifying ideal sentence pairs and passing useless ones
# to speed up the training process and keep the model as clean as possible
regexG = re.compile(r'\s*<tu>\s*<tuv xml:lang="ar"><seg>.*[\u0600-\u06FF]+.*
                    </seg></tuv>\s*<tuv xml:lang="en"><seg>.*\s*.*</seg></tuv>\s*</tu>')
```

Figure 2.4: Sentence pair's regular expression pattern.

2.4.3 Feeding GenSim

Right before starting the training, GenSim's input should be in a *list of words lists* shape, the global list corresponds to the trained document, or part of document, the inner lists correspond to the sentences. The only difference relies on the shuffling Modes on sentence level which we will discuss in the next section.

2.5 Training

Our training experiments were based on finding an appropriate alignment method on sentence level regarding the thorough difference between Arabic and English.

2.5.1 Parallel Mode

we prefaced our empirical procedure with making the sentence pairs preserve their words order and put them next to each other as they are a single sentence, the following is an example of document with one sentence as figure 2.5 shows.

```
[['الكلية' ,'يغادر' ,'محمد', 'mohammed', 'leaves', 'faculty'],
 ['القدم' ,'كرة' ,'يلعب' ,'يوسف', 'youcef', 'plays', 'football']]
```

Figure 2.5: Parallel Mode example.

2.5.2 Word by Word Mode

The idea behind this alignment was making the word and its translation close to each other, hence, they will be captured together by *Word2Vec* context window as shown in figure 2.6.

```
[['mohammed', 'محمد', 'leaves', 'يغادر', 'faculty', 'الكلية'],
 ['يوسف', 'youcef', 'يلعب', 'plays', 'كرة', 'football', 'القدم']]
```

Figure 2.6: Word by word Mode example.

2.5.3 Random Shuffle Mode

This mode shares the same idea with the precedent one, which is getting translation pairs closer hopefully in the same context window. So we randomly shuffled each sentence pair separately to have as figure 2.7 shows.

```
[['محمد', 'leaves', 'mohammed', 'الكلية' ,'يغادر', 'faculty'],
 ['plays', 'يوسف', 'youcef', 'يلعب', 'football', 'كرة' ,'القدم']]
```

Figure 2.7: Random shuffle Mode example.

2.5.4 Training Code

Here we provide the code we developed to train our model variants with a different function to each mode and a global tmx reader and trainer function.

```
# extract sentence pairs from corpus file & train them
def tmxTrainer(tmxFileLocation, modelLocation, re_train, stopped_count = 0):

  print("scouting & planning...")
  with open(tmxFileLocation, 'r') as f:
    # lines from 29 to 49 are for identifying the number of pairs
    # and getting rid of first useless identification lines on top of tmx file

    regexG = re.compile(r'<tu>')
    line = f.readline()
    count = 0
    for i in range(100):
      if regexG.search(line) == None:
        count += 1
        line = f.readline()
      else:
        break

    f.seek(0)
    for i, line in enumerate(f):
      pass

    pairsNumber = int(((i + 1) - count)/4)
    print("pairs to be trained = {}".format(pairsNumber))

    print("getting rid of useless lines...")
    f.seek(0)
    for i in range(0, count):
      f.readline()

    # regexG pattern is used for identifying ideal sentence pairs and passing useless ones
    # to speed up the training process and keep the model as clean as possible
    regexG = re.compile(r'\s*<tu>\s*<tuv xml:lang="ar"><seg>.*[\u0600-\u06FF]+.*
                        </seg></tuv>\s*<tuv xml:lang="en"><seg>.*\s*.*</seg></tuv>\s*</tu>')

    print("begin training corpus")

    # when colab runtime unexpectedly disconnects...the last index can be retrieved from output
    # and affected to stopped_count argument so the training can restart where it left off (simply by passing already trained pairs)
    for i in range(0,stopped_count):
      chunk = "{}\n{}\n{}\n{}".format(f.readline(), f.readline(), f.readline(), f.readline())


    documents = []
    start = time.time()
     for i in range(0, 33000):
       chunk = "{}\n{}\n{}\n{}".format(f.readline(), f.readline(), f.readline(), f.readline())
       if regexG.search(chunk) == None:
         pass
       else:
         documents.append(simple_preprocess(stopWordsRemover(sentencePairReader(chunk))))


     if re_train == 0:
       print("creating model")
       model = Word2Vec(documents, size = 300, window = 5, min_count = 10, workers = 4, sg = 0)
       model.save(modelLocation)
       print("sentence {}: model intialized and trained on first 33000 sentence pairs, vocab now holds {} words".format(i + 1 + stopped_count, len(model.wv.vocab)))

     elif re_train == 1:
       print("loading model")
       model = Word2Vec.load(modelLocation)
       model.build_vocab(sentences = documents, update = True)
       model.train(documents, total_examples = len(documents), epochs = 10)
       model.save(modelLocation)
       print("sentence {}: model loaded and trained on other 33000 sentence pairs, vocab now holds {} words".format(i + 1 + stopped_count,len(model.wv.vocab)))

     documents = []
     for i in range(0, pairsNumber - 33000 - stopped_count):
       chunk = "{}\n{}\n{}\n{}".format(f.readline(), f.readline(), f.readline(), f.readline())
       if regexG.search(chunk) == None:
         pass
       else:
         documents.append(simple_preprocess(stopWordsRemover(sentencePairReader(chunk))))
         if len(documents) == 33000:
           model.build_vocab(sentences = documents, update = True)
           model.train(documents, total_examples = len(documents), epochs = 10)
           model.save(modelLocation)
           print("sentence {}: model trained other 33000 sentence pairs, vocab now holds {} words".format(i + 33000 + stopped_count, len(model.wv.vocab)))
           documents = []

     model.build_vocab(sentences = documents, update = True)
     model.train(documents, total_examples = len(documents), epochs = 10)
     model.save(modelLocation)
     print("sentence {}: model trained other 33000 sentence pairs, vocab now holds {} words".format(i + 33000 + stopped_count, len(model.wv.vocab)))
     end = time.time()


  print("DONE :)")
  print("time spent in traning (in seconds): {}".format(end - start))
```

Figure 2.8: Training code of one of the variants.

2.6 Model Variants Evaluation

Based on the difference between the alignment methods we used, model variants should vary in their given results, and usually Cross-lingual models get evaluated concerning both mono-lingual and cross-Lingual aspects. We will discuss these matters while we intrinsically evaluate our models.

2.6.1 Intrinsic Evaluation

There are a lot of evaluation ways in NLP that we can name *"intrinsic"*, maybe the famous or at least widely used ones would be *questions answering* (mainly aims to investigate wether the model preserves relationships between tokens) and *lexical translation* (by trying to show the model's ability to translate tokens) which is also what we will exercise to evaluate our variants.

2.6.1.1 Lexical Translation task

In general, this task means evaluating the model in way that analyzes its ability in capturing translation relationship between words Gouws et al. (2015). This could be done in many ways. Therefore, we did improvise and made our evaluation procedure.

```
1  def isArabic(word):
2    regex = re.compile(r'[\u0600-\u06FF]+')
3    if regex.search(word) == None:
4      return False
5    else:
6      return True
7
8  def tuplesConstructer(vocabulary):
9    tuples = []
10   for i in range(1000):
11     for j, s in enumerate(model.wv.most_similar(vocabulary[i])):
12       if isArabic(s[0]) != isArabic(vocabulary[i]):
13         tuples.append((vocabulary[i], s[0]))
14
15   return tuples
```

Figure 2.9: Code of constructing word pairs.

The evaluation was based on two main operations:

- **Extract pairs:** by interacting with the model and calling for the most similar word of a randomly chosen word with one condition, similar word should be in the opposite language. For that we varied our evaluation on multiple *top_k* similarities, that means giving a limited range for that opposite language word to occur in similarities list, extracting word couples code is illustrated in figure 2.9.
- **Compare to Google Translate API:** by running our extracted translation against **Translate API** bag of words of the origin word translation.

The comparison operation was processed manually, hence we worked on 1000 word pairs to evaluate. For every positive sample we add a 0.001 to a *sum* variable initialized with 0. In the end the score of the model will be calculated as: $score = (sum/1000) * 100$.

Starting with **Parallel Mode**, its results has nothing in common with those of the others, its cross-lingual aspect was completely off but mono-lingually it was as if two models of different languages were separately trained, it would be so logical for a model like so to give high scores in questions answering and other ruined in lexical translation which is exactly what our evaluation came up with. On the other hand,

Word by word Mode with CBOW didn't have much of attraction in both aspects, but SkipGram had a good improvement for lexical translation. Concerning **Random shuffle Mode**, its variants gave the best lexical translation scores, especially SkipGram where translation pairs can clearly appear next to or on top of each other when we use *t-Distributed Stochastic Neighbor Embedding* (t-SNE) Maaten and Hinton (2008) and *Matplotlib* to visualize our data as figure 2.11 shows. Table 2.1 gathers all model variants scores.

	CBOW					Skip-Gram				
#Modes	**Top1**	**Top2**	**Top3**	**Top5**	**Top10**	**Top1**	**Top2**	**Top3**	**Top5**	**Top10**
Parallel	0.1%	0.5%	0.7%	1.2%	2.1%	2.8%	4.5%	6.1%	6.1%	9.3%
W. by W.	4.1%	11.3%	17.4%	25.3%	37.2%	60.6%	73.5%	78.3%	86.8%	92.4%
Random	57.7%	71.4%	79.2%	85.3%	90.5%	62.4%	74.2%	78.4%	87.5%	**93.8%**

Table 2.1: Lexical Translation evaluation results of ArbEngVec models

Discussion: We evaluate our model by assuring its ability to recognize that the most similar word from Y language is the translation of a given word from X language. We then widen our evaluation by replacing the most similar word with K-most similar list of words. Now the results are essentially based on the context window, hence, a failed variant such as **Parallel-Skip** means translation pairs were far from being in the same context. In the other hand, **Random-Skip** giving the best result means it is the best alignment choice for now to capture translation pairs.

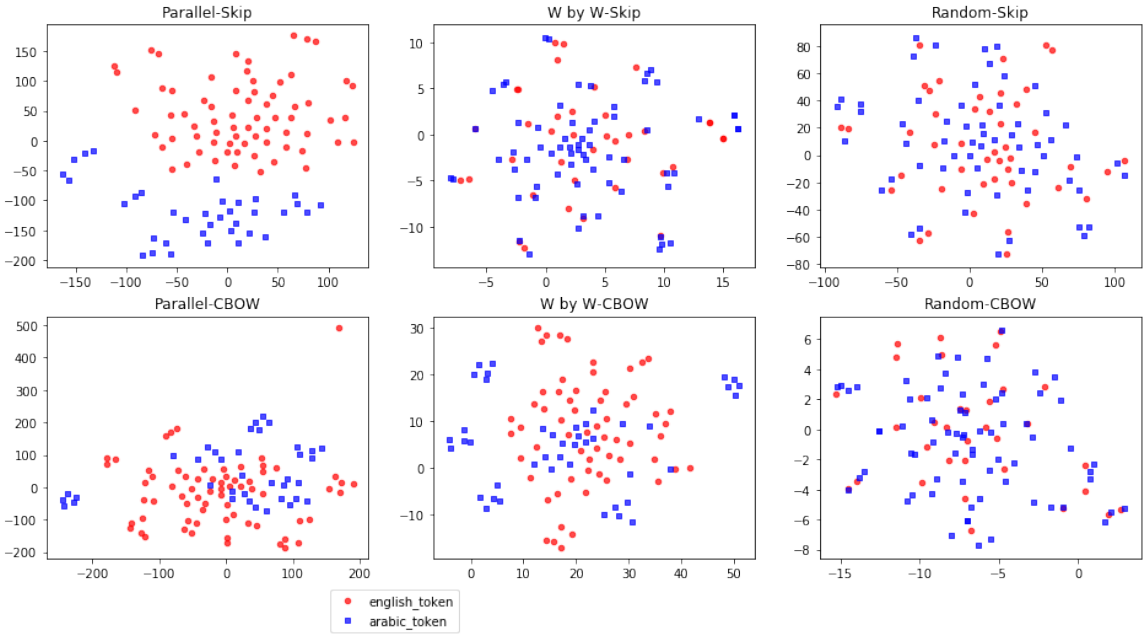

Figure 2.10: ArbEngVec variants visualization.

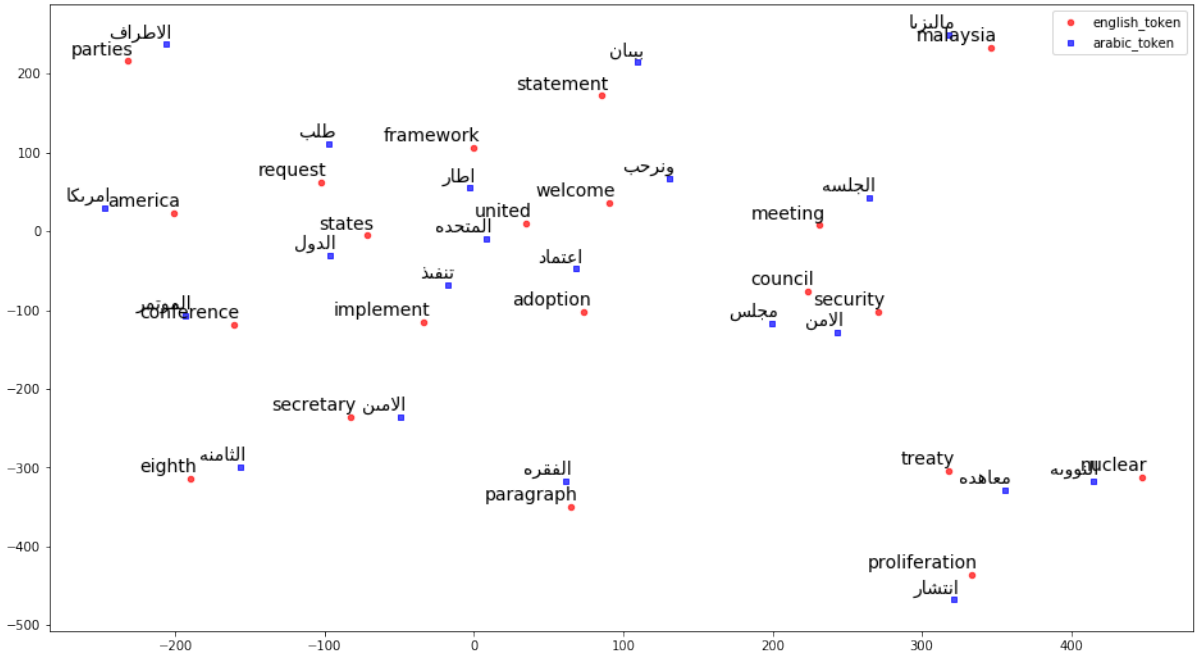

Figure 2.11: Random SkipGram model Visualization.

The model variants that gave us high scores in lexical translation were the same that they gave back skewed questions answering results. Those variants have something in common which is the lack of words meaningful order after shuffle or alignment, questions answering depends on words relationships preservation and relationships cannot be preserved anymore since the distance between examined words varies from a sample to an other.

Principal contribution

It is worth mentioning, that we managed to publish this chapter work specifically as a scientific paper in *WANLP 2019* conference [12], co-located with *ACL 2019* (see Appendix 3.4.1).

Conclusion

In this chapter, we provided many Arabic-English cross-lingual word embedding model variants which wen named ***ArbEngVec***. They were learned through a large dataset collection of parallel Arabic-English sentences. Also, we evaluated the ArbEngVec variants via intrinsic evaluation using Word Translation task.

[12] paper is declared as accepted: https://sites.google.com/view/wanlp-2019/home?authuser=0

Chapter 3

Cross-Lingual Semantic Similarity System

"We could convert a news article about a company into a number that expresses positive sentiment...and negative sentiment."

Liang Zhou

Introduction

An other major formal evaluation task for word embeddings is the Extrinsic Evaluation, which is based on surveilling the model under its usage as core for real-world NLP problem solving systems. Our choice fell on *Cross-Lingual Semantic Textual Similarity* (CL-STS). In this chapter's first section 3.2 we will present the mere similarity system and its application. In section 3.3 will cover a combination between the mere system and many weighting approaches. Finally section 3.4 shows the result of our system against the human annotation of sentence pairs.

3.1 System Global Schema

In figure 3.1 we describe the most important parts of our system. At the upper part we have external plugins including our model and two other weighting structures. At the lower part, we start with both inputs Arabic and English sentences, Arabic Preprocessor where unneeded characters are deleted such as diacritics, Global Preprocessor where digits are removed, then after Tokenization we assign to every word its appropriate vector, weighting is optional (to discuss in subsequent sections). Finally, we sum up vectors and provide two global sentence vectors to calculate *cosine* distance between them.

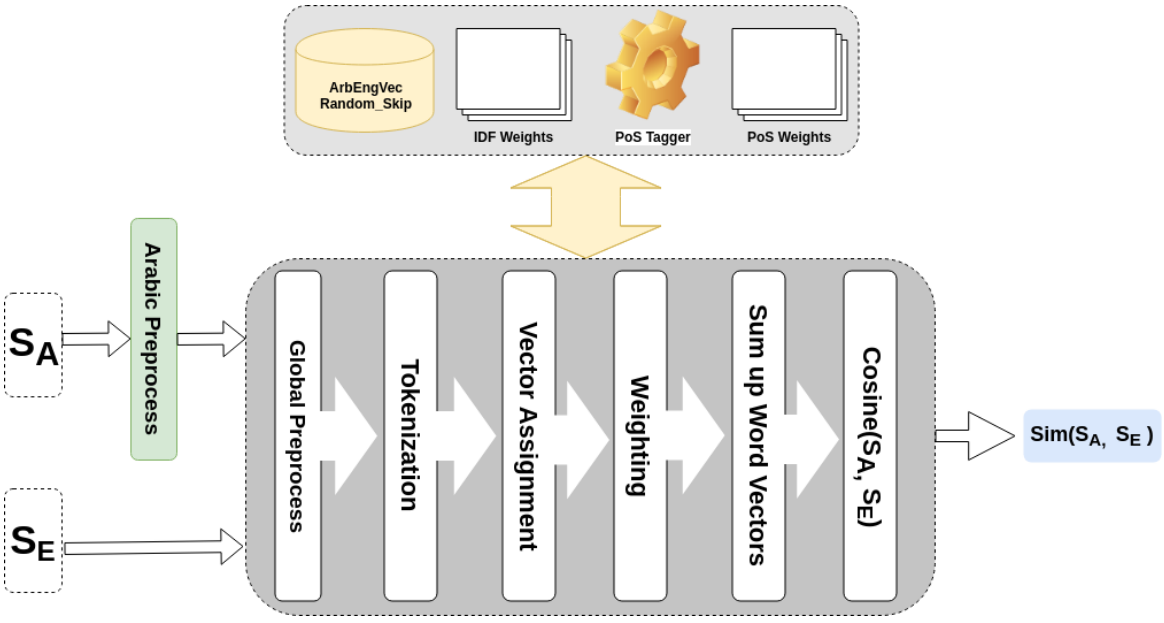

Figure 3.1: Semantic Similarity System, Global Schema.

3.2 Sentences Similarity

The simplest way any one could think of in evaluating similarity between sentence pairs is summing up word vectors of every sentence providing by that two new sentence vectors and calculate how distant are they using *cosine_similarity*, code of this procedure is illustrated in figure 3.3. This intuitive way has one major inconvenience. If we suppose that both sentences have many different unrelated words, after summing up their vectors this would get us to very similar sentence vectors, hence, giving a high similarity score which is not the case.

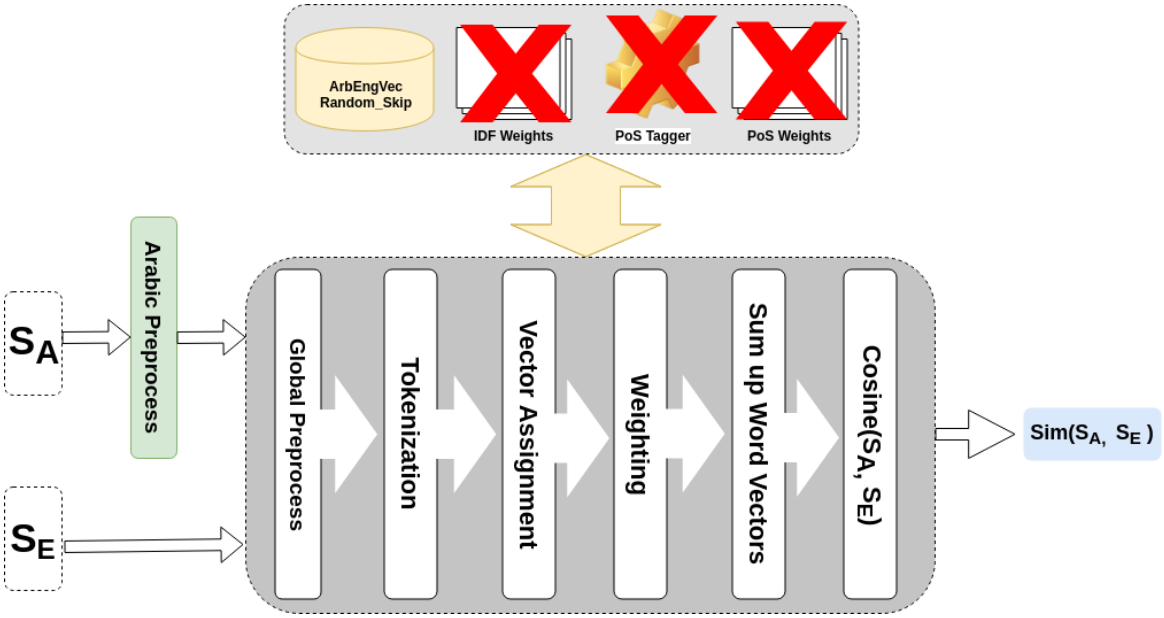

Figure 3.2: Mere Similarity System architecture.

```
# mere similarity evaluation
def mereCosineSentencesSimilarity(s_1, s_2):
  s_1 = simple_preprocess(s_1)
  s_2 = simple_preprocess(s_2)

  V_1 = np.zeros(model.wv['malaysia'].shape)
  V_2 = np.zeros(model.wv['malaysia'].shape)


  for word in s_1:
    if word not in model.wv.vocab:
      pass
    else:
      V_1 += model.wv[word]

  for word in s_2:
    if word not in model.wv.vocab:
      pass
    else:
      V_2 += model.wv[word]

  sim = 1 - spatial.distance.cosine(V_1, V_2)

  return sim
```

Figure 3.3: Code of simple cosine similarity.

3.3 Weighted Sentence Similarity

The importance of words differs based on the word nature and its position in the sentence. Exactly as Nagoudi and Schwab did Schwab et al. (2017), we are going to weigh every word based on its position in the sentence and its occurrence in the corpus.

3.3.1 TF-IDF Weighting:

tf-idf focuses on assigning different factors for different words of the built-during-training vocabulary. Each word has its *tf-idf* weight and in general it corresponds to this word's usage frequency in the trained corpus Ramos et al. (2003).

What we did first is providing the two most important measures (figure 3.5 is the code of extracting them):

1. Number of documents that a word occur in.
2. Occurrence of this word in each document.

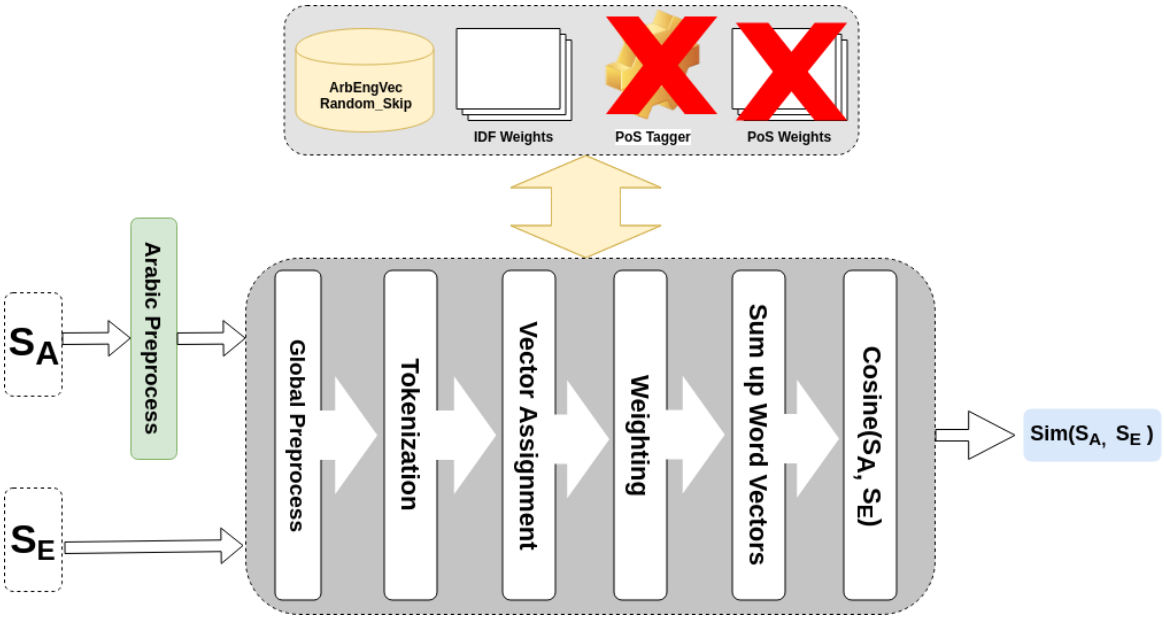

Figure 3.4: Similarity System with IDF architecture.

```
def tf_idfCounter(document, docCount, vocabulary):
  corpusIterate = tmxReader(document)
  for pair in corpusIterate:
    for k, sentence in enumerate(pair.split("\t")):
      if k == 0:
        sentence = arabic_preprocesser(sentence)
      sentence = simple_preprocess(sentence)
      for word in sentence:
        if word in vocabulary:
          vocabulary[word][docCount] += 1

  return vocabulary
```

Figure 3.5: Extracting tf-idf measures.

```
myVocab = {}

with open("tf_idf.csv", 'r') as old_f:
  csvRead = csv.DictReader(old_f, delimiter = "\t")
  for line in csvRead:
    myVocab[line['word']] = [int(s) for s in line['tfs'].strip('[]').split(',')]

print(myVocab['parties'])
```

```
myFs = ["ar-en_3.tmx", "ar-en_4.tmx", "ar-en_5.tmx", "ar-en_6.tmx", "ar-en_7.tmx"



for docCount, file in enumerate(myFs):
  print("{} ###################################".format(file))
  myVocab = tf_idfCounter(os.path.join("corpus", file), docCount, myVocab)
  with open("tf_idf.csv", 'w') as f:
    csvWrite = csv.DictWriter(f, fieldnames = ['word', 'tfs'], delimiter = '\t')

    csvWrite.writeheader()

    for word in myVocab:
      tmpDict = {}
      tmpDict ['word'] = word
      tmpDict ['tfs'] = myVocab[word]
      csvWrite.writerow(tmpDict)

print("done saving weights")
```

Figure 3.6: Get across whole corpus to provide tf-idf weights.

3.3.2 Part of Speech Tags Weighting:

The importance of a word varies with the nature of the sentence it is used in. For instance, a verb in a verbal phrase is more critical than a noun, same thing goes for a noun in a nominal phrase. Due to that, we went through tagging Arabic sentences with Gahbiche-Braham et al. (2012) Part of Speech (PoS) tagger tool in order to eventually give each token an appropriate weight. For English sentences, we used the built in *Natural Language ToolKit* (NLTK) Python library, specifically *pos_tag* function to do the same. The code of tagging sentences is provided in figures 3.8 and 3.9.

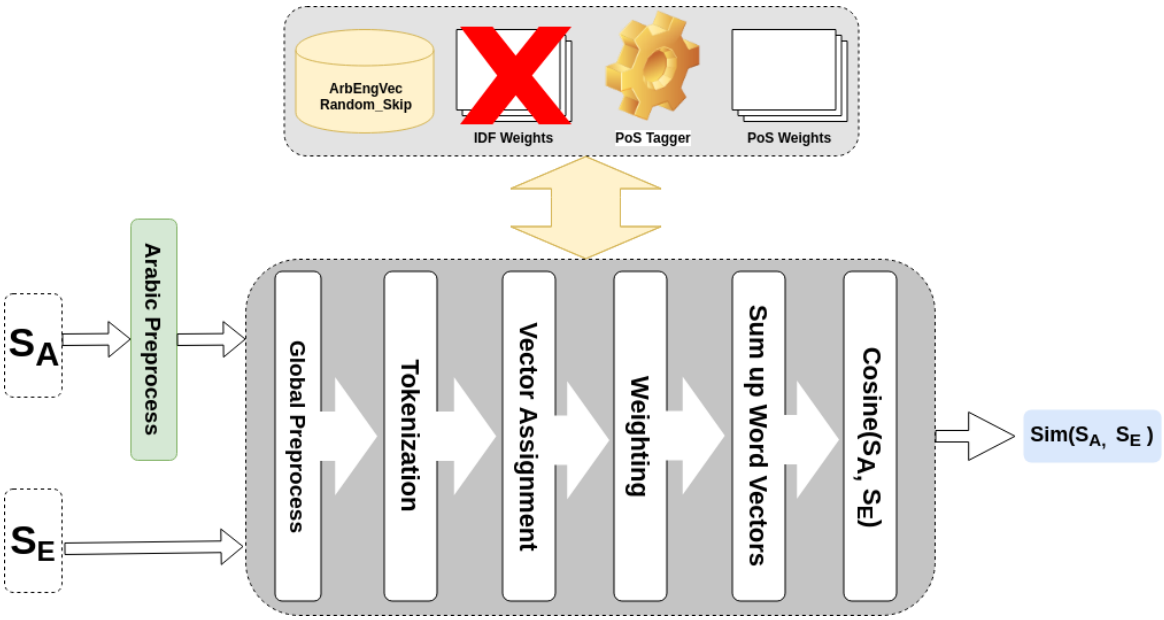

Figure 3.7: Similarity System with PoS architecture.

```
sapaPath = "/usr/local/bin/Wapiti/SAPA"
projectPath = "/home/raki22/Desktop/CL-STS"


with open("annotated_corpus.csv", 'r') as csv_f:
    csvRead = csv.DictReader(csv_f, delimiter = '\t')
    os.chdir(sapaPath)
    with open("exapo.txt", 'w') as exapo:
        lines = [line['arabic'] for line in csvRead]
        for line in lines:
            exapo.write("{}\n".format(line))

    os.system("sudo {}".format("./segment.sh exapo.txt"))
    os.chdir(projectPath)
    with open(os.path.join(sapaPath,"exapo.txt.result"), 'r') as f:
        with open(os.path.join(sapaPath,"exapo.txt.norm.pos"), 'r') as ff:
            with open("corpus_pos.csv", 'w') as csv_pos:
                csvWrite = csv.DictWriter(csv_pos, fieldnames = ['ar_sentence', 'ar_pos']

                csvWrite.writeheader()

                results = f.read().split('\n')
                norms = ff.read().split('\n')

                for xl in range(len(norms)):
                    tmpDict = {}
                    tmpDict['ar_sentence'] = results[xl]
                    tmpDict['ar_pos'] = norms[xl]
```

```
def arabicPoSTagger():
    tagsWeights = {'verb': 0.82, 'noun': 0.97, 'adj': 0.85, 'conj': 0.08, 'punc': 0.19

    # with open(os.path.join(sapaPath,'exapo.txt'), 'w') as f:
    #   f.write(sentence)
    # os.chdir(sapaPath)
    # os.system("sudo {}".format("./segment.sh exapo.txt"))
    # os.chdir(projectPath)
    # with open(os.path.join(sapaPath,"exapo.txt.result"), 'r') as f:

    with open("corpus_pos.csv", 'r') as c_p:
        csvRead = csv.DictReader(c_p, delimiter = '\t')

        buckwalterSentences = []
        posS = []

        for line in csvRead:
            buckwalterSentences.append(line['ar_sentence'])
            posS.append(line['ar_pos'])

        for bu, buckwalterWords in enumerate(buckwalterSentences):
            taggedSentence = {}
            buckwalterWords = buckwalterWords.split()
            poss = posS[bu].split()

            for i, token in enumerate(buckwalterWords):
                regex = re.compile(r'.{1}\+')
                if regex.search(token) != None:
                    token = token + " " + buckwalterWords[i + 1]
```

```
42                  token = token + " " + buckwalterWords[i + 1]
43                  try:
44                      taggedSentence[arabic_preprocesser(BuckWalter_Transliterator(token, 1))]
45                  except KeyError:
46                      taggedSentence[arabic_preprocesser(BuckWalter_Transliterator(token, 1))]
47              else:
48                  try:
49                      taggedSentence[arabic_preprocesser(BuckWalter_Transliterator(token, 1))]
50                  except KeyError:
51                      taggedSentence[arabic_preprocesser(BuckWalter_Transliterator(token, 1))]
52
53              if regex.search(buckwalterWords[i - 1]) == True:
54                  continue
55
56          yield taggedSentence
```

Figure 3.8: Tag Arabic sentences using Gahbiche-Braham et al. (2012) Segmentor And Part-of-speech tagger for Arabic (SAPA) tool.

```
60  def enPoSTagger(sentence):
61      tagsWeights = {'VB': 0.82, 'NN': 0.97, 'JJ': 0.85, 'TO': 0.06, 'CD': 0.05, 'CC': 0.05
62      taggedSentence = {}
63      regex = re.compile(r'(VB.*|NN.*|JJ.*)')
64      tmp = word_tokenize(sentence)
65      pos_words = pos_tag(tmp)
66
67      for token in pos_words:
68          if regex.search(token[1]) != None:
69              pos = token[1][0:2]
70              taggedSentence[token[0].lower()] = tagsWeights[pos]
71          else:
72              try:
73                  taggedSentence[token[0].lower()] = tagsWeights[token[1]]
74              except KeyError:
75                  taggedSentence[token[0].lower()] = 0.5
76
77      return taggedSentence
```

Figure 3.9: Tag English sentences using nltk.pos_tag function.

3.3.3 IDF and PoS Combined:

In this part, we used $POS\&IDF$ weighting method proposed by *Nagoudi et al.*Nagoudi et al. (2017) to see if the performance of our WE model gets improved with it. The code of assigning combined weights is in figure 3.10.

```
19      if weighting == 'idf_pos':
20          posDict = {**next(posS), **enPoSTagger(s_2)}
21
22          def weighter(word, dict = posDict):
23              try:
24                  return dict[word] * idfAffecter(word)
25              except KeyError:
26                  return 0.5 * idfAffecter(word)
```

Figure 3.10: Assign combined weights to words.

After adding weighting method, surely *cosine_similarity* function would have some changes, figure 3.11 contains the new function script.

```
posS = arabicPoSTagger()
posS_2 = arabicPoSTagger()
# mere similarity evaluation
def mereCosineSentencesSimilarity(s_1, s_2, weighting = None):
  weightDict = {None: oneWeighter, 'idf': idfAffecter}

  if weighting == 'idf_pos':
    posDict = {**next(posS), **enPoSTagger(s_2)}

    def weighter(word, dict = posDict):
      try:
        return dict[word] * idfAffecter(word)
      except KeyError:
        return 0.5 * idfAffecter(word)

  elif weighting == 'pos':
    posDict = {**next(posS_2), **enPoSTagger(s_2)}

    def weighter(word, dict = posDict):
      try:
        return dict[word]
      except KeyError:
        return 0.5
  else:
    weighter = weightDict[weighting]

    s_1 = simple_preprocess(s_1)
    s_2 = simple_preprocess(s_2)

    V_1 = np.zeros(model.wv['malaysia'].shape)
    V_2 = np.zeros(model.wv['malaysia'].shape)


    for word in s_1:
      if word not in model.wv.vocab:
        pass
      else:
        V_1 += model.wv[word] * weighter(word)

    for word in s_2:
      if word not in model.wv.vocab:
        pass
      else:
        try:
          V_2 += model.wv[word] * weighter(word)
        except KeyError:
          continue

    sim = 1 - cosine(V_1, V_2)

    return sim
```

Figure 3.11: Cosine Similarity function after adding weighting method.

3.4 Results

3.4.1 Pearson Correlation:

Pearson Correlation is a coefficient calculated for measuring how strong a relationship between two sets of values can be and to which direction. Meaning, is the relationship between these two value groups *directly proportional* or *inversely proportional* and how powerful is it. Both mentioned infos can be captured from the coefficient value that varies between 1 and −1 which they mean directly and inversely proportional respectively and where 0 is translated as *no_correlation* Bolboaca and Jäntschi (2006).

We evaluated our multiple system scores with Pearson Coefficient by running every score separately against human score using more than 600 sentence pairs from Nagoudi et al. (2017) annotated corpus. Table 3.1 contains the evaluation of our best model *Random Shuffle Skipgram* with and without weights.

mere system	system with idf	system with pos	idf and pos combined
76.55%	74.96%	72.89%	70.84%

Table 3.1: Pearson Correlation for System Scores against Human Score

Discussion:

As Table 3.1 has shown, an abstract system without any weights gave better Pearson score. While different weights has only made it worse and the worst happened when combining both of weights. These results should be considered awkward because weights often , if not always, give better results when plugged with a semantic similarity system. This decline in results is due to two different facts. First, for *idf* when translation pairs are not being aligned, hence not given equal coefficients across the languages, this kind of weighting will surely dampen the similarity score. Second, assigning PoS weights to tokens is useful when they are trained on an ordered meaningful context and obviously not in shuffled ones where clearly vector representation would loose a considerable amount of context features.

Conclusion

In this chapter we took a step further in evaluating our WE model and that was through Extrinsic Evaluation. We constructed the mere system which was based completely on *ArbEngVec* word representation vector space. Then, we provided different weights, each one of them revives the importance of one or more of the linguistic aspects. We evaluated all system variants against human score using an annotated corpus by applying Pearson Correlation separately with each result.

Conclusion

There are multiple ways for providing Natural Language Processing tools with a suitable core. Word Embeddings is more common than any other concept. In this thesis we provided an exhaustive study about Word Embeddings with both Mono and Cross-Lingual aspect.

After that, we collected a large textual dataset, preprocessed the text and normalized it. Then, we presented our contribution which was a Cross-Lingual Word Embedding model evaluated by Lexical Translation task against Google Translate API where the results were so promising (see Appendix 3.4.1 to check out the accepted paper).

In the final chapter, we evaluated our WE across a real-world NLP application and that is Semantic Textual Similarity as an Extrinsic Evaluation approach. The abstract system has given a good score and this latter started to diminish as we plugged in some weighting methods, hence we managed to focus the light on our system inconveniences and therefore learn exactly what we have to improve, add and replace in our proposal as a future work. We actually highlighted the alignment method and made it number one priority to be replaced with a better one.

Bibliography

Bachchhav, K. P. (2016). Information retrieval: search process, techniques and strategies. *International Journal of Next Generation Library and Technologies*, 2(1):7.

Baeza-Yates, R., Ribeiro-Neto, B., et al. (1999). *Modern information retrieval*, volume 463. ACM press New York.

Bolboaca, S.-D. and Jäntschi, L. (2006). Pearson versus spearman, kendall's tau correlation analysis on structure-activity relationships of biologic active compounds. *Leonardo Journal of Sciences*, 5(9):179–200.

Carneiro, T., Da Nóbrega, R. V. M., Nepomuceno, T., Bian, G.-B., De Albuquerque, V. H. C., and Reboucas Filho, P. P. (2018). Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, 6:61677–61685.

Chen, X., Sun, Y., Athiwaratkun, B., Cardie, C., and Weinberger, K. (2018). Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. pages 160–167.

Creutz, M. (2018). Open subtitles paraphrase corpus for six languages. *arXiv preprint arXiv:1809.06142*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Eisele, A. and Chen, Y. (2010). Multiun: A multilingual corpus from united nation documents.

Ferrero, J., Agnes, F., Besacier, L., and Schwab, D. (2017). Usingword embedding for cross-language plagiarism detection. *arXiv preprint arXiv:1702.03082*.

Gahbiche-Braham, S., Bonneau-Maynard, H., Lavergne, T., and Yvon, F. (2012). Joint segmentation and pos tagging for arabic using a crf-based classifier. In Chair), N. C. C., Choukri, K., Declerck, T., Dogan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proc. of LREC'12*, pages 2107–2113, Istanbul, Turkey. European Language Resources Association (ELRA).

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Gouws, S., Bengio, Y., and Corrado, G. (2015). Bilbowa: Fast bilingual distributed representations without word alignments.

Joshi, N. and Mathur, I. (2012). Design of english-hindi translation memory for efficient translation. *arXiv preprint arXiv:1210.5517*.

Khorsi, A., Cherroun, H., Schwab, D., et al. (2018). A two-level plagiarism detection system for arabic documents. *Cybernetics and Information Technologies*, 20.

Kusner, M., Sun, Y., Kolkin, N., and Weinberger, K. (2015). From word embeddings to document distances. pages 957–966.

Levy, O. and Goldberg, Y. (2014a). Dependency-based word embeddings. 2:302–308.

Levy, O. and Goldberg, Y. (2014b). Neural word embedding as implicit matrix factorization. pages 2177–2185.

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Luong, T., Pham, H., and Manning, C. D. (2015). Bilingual word representations with monolingual quality in mind. pages 151–159.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Hlt-naacl*, volume 13, pages 746–751.

Mnih, A. and Hinton, G. E. (2009). A scalable hierarchical distributed language model. pages 1081–1088.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. 5:246–252.

Nagoudi, E. M. B., Ferrero, J., Schwab, D., Cherroun, H., et al. (2017). Word embedding-based approaches for measuring semantic similarity of arabic-english sentences. In *International Conference on Arabic Language Processing*, pages 19–33. Springer.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. pages 1532–1543.

Rafalovitch, A., Dale, R., et al. (2009). United nations general assembly resolutions: A six-language parallel corpus. 12:292–299.

Ramos, J. et al. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ.

Řehřek, R. and Sojka, P. (2011). Gensim—statistical semantics in python. *statistical semantics; gensim; Python; LDA; SVD*.

Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Ruder, S., Vulić, I., and Søgaard, A. (2017). A survey of cross-lingual word embedding models. *arXiv preprint arXiv:1706.04902*.

Schwab, D. et al. (2017). Semantic similarity of arabic sentences with word embeddings. pages 18–24.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Tiedemann, J. (2012). Parallel data, tools and interfaces in opus. 2012:2214–2218.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. pages 384–394.

Vulić, I. and Moens, M.-F. (2015). Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. pages 363–372.

Wang, W., Zhao, T., and Zhang, C. (2013). Bilingual seed lexicon adaptation for entity translation extraction. pages 1309–1313.

Wołk, K. and Marasek, K. (2014). Building subject-aligned comparable corpora and mining it for truly parallel sentence pairs. *Procedia Technology*, 18:126–132.

Yinfei Yang, C. T. (2018). Advances in semantic textual similarity. https://ai.googleblog.com/2018/05/advances-in-semantic-textual-similarity.html.

Zahran, M. A., Magooda, A., Mahgoub, A. Y., Raafat, H., Rashwan, M., and Atyia, A. (2015). Word representations in vector space and their applications for arabic. pages 430–443.

ZarrabiZadeh, H. (2007). Tanzil project. *URL: http://tanzil. net/wiki/Tanzil_Project*.

Ziemski, M., Junczys-Dowmunt, M., and Pouliquen, B. (2016). The united nations parallel corpus v1. 0. In *Lrec*.

Appendix

In this Appendix we will provide the published paper, the one mentioned in Principal Contribution section, Chapter 2.

ArbEngVec : Arabic-English Cross-Lingual Word Embedding Model

Raki Lachraf
Echahid Hamma Lakhdar University,
El Oued, Algeria
raki.lachraf@univ-eloued.dz

El Moatez Billah Nagoudi
Echahid Hamma Lakhdar University,
El Oued, Algeria
LIM laboratory, Laghouat
moatez-nagoudi@univ-eloued.dz

Youcef Ayachi
Echahid Hamma Lakhdar University
El Oued, Algeria
youcef.ayachi@univ-eloued.dz

Ahmed Abdelali
Hamad Bin Khalifa University
Qatar Computing Research Institute
Doha, Qatar
aabdelali@qf.org.qa

Didier Schwab
LIG-GETALP
Univ. Grenoble Alpes,
France
didier.schwab@imag.fr

Abstract

Word Embeddings (WE) are getting increasingly popular and widely applied in many Natural Language Processing (NLP) applications due to their effectiveness in capturing semantic properties of words; Machine Translation (MT), Information Retrieval (IR) and Information Extraction (IE) are among such areas. In this paper, we propose an open source ArbEngVec which provides several Arabic-English cross-lingual word embedding models. To train our bilingual models, we use a large dataset with more than 93 million pairs of Arabic-English parallel sentences. In addition, we perform both extrinsic and intrinsic evaluations for the different word embedding model variants. The extrinsic evaluation assesses the performance of models on the cross-language Semantic Textual Similarity (STS), while the intrinsic evaluation is based on the Word Translation (WT) task.

1 Introduction

Distributed word representations in vector space (Word Embeddings) are one of the most successful applications in deep learning for capturing the semantic and syntactic properties of words. Lately, many NLP tasks have been enriched using tools based on Mono and Cross-Lingual word embedding models. For instance, Mono-Lingual Word Embeddings (MLWE) have been widely used in information retrieval (Vulić and Moens, 2015a), sentiment analysis (Tang et al., 2014; Nagoudi, 2018) text classification (Lai et al., 2015), semantic textual similarity (Kenter and De Rijke, 2015; Nagoudi and Schwab, 2017) and plagiarism detection (Nagoudi et al., 2018).

Cross-Lingual Word Embeddings (CLWE) is a more challenging task because the knowledge is transferred between two or more different languages (Doval et al., 2018). Recently, cross-lingual word embeddings was used to address several issues, e.g. machine translation (Zou et al., 2013), cross-language information retrieval (Vulić and Moens, 2015a; Zhou et al., 2012), cross-language semantic similarity (Ataman et al., 2016; Nagoudi et al., 2017b) and plagiarism detection across multiple languages (Ferrero et al., 2017; Barrón-Cedeño et al., 2013). Many cross-lingual word embedding models in natural language have been developed, particularly for English, but Arabic did not get that much of interest.

In this paper, we propose six Arabic-English cross-lingual word embedding models[1]. To train these models, we have used a large collection with more than 93 million pairs of parallel Arabic-English sentences.

The rest of this paper is organised as follows: in section 2 we provide a quick overview of work related to the cross-lingual word embedding models. We describe our dataset collection and the preprocessing process in Section 3. Section 4 presents our proposed cross-lingual models. Section 5 presents the evaluation results. Section 6 concludes the paper with our main findings and points to possible directions for future work.

2 Related works

While we focus on the cross-lingual word embedding models, the interested reader may refer to a number of research studies on the subject of mono-lingual word embeddings in general (Collobert and Weston, 2008), (Turian et al.,

[1] All models can be downloaded from : https://github.com/Raki22/ArbEngVec.git

2010), (Mnih and Hinton, 2009), (Mikolov et al., 2013c,b) and (Peters et al., 2018).

In the cross-lingual context, several word embedding models are proposed. Blunsom and Hermann (2014) introduced a Bilingual Compositional Model (BiCVM). Leveraging from the fact that aligned sentences have the same meaning. BiCVM is based on a sentence-aligned corpus to learn the bilingual word embedding vectors.

Vulić and Moens (2015b) introduced a Bilingual Word Embedding Skip-Gram (BWESG), this model is constructed through three main steps: i) prepare a Skip-Gram Negative Sampling (Mikolov et al., 2013b) architecture that deals with document aligned comparable data, ii) provide bilingual document pairs, iii) shuffle each pair producing pseudo-bilingual document that serves as the architecture's input which is to be trained.

Luong et al. (2015) proposed a Bilingual Skip-Gram model (BiSKip). BiSKip uses the Skip-Gram of (Mikolov et al., 2013b) to train two different languages at the same time by manipulating the Skip-Gram architecture to obtain two pivots and two contexts and provide a training session for each combination. Choosing two Germanic languages (English and German) made it easier to predict target language's appropriate pivot and context for the ones from source language by simply aligning the target words at position $[i * T/S]$ with source words at position i where S and T are source and target sentence lengths respectively.

Chen et al. (2018) presented an Adversarial Deep Averaging Network (ADAN) for cross-lingual sentiment classification. In fact, they trained many bilingual WE models, one of them was trained using the United Nations (UN) English-Arabic parallel aligned corpus (Ziemski et al., 2016) and Bilingual Bag-of-Words without Alignments (BilBOWA) (Gouws et al., 2015). Additionally, ADAN replaces the softmax and regularization terms by a less costly alternatives.

Recently, Devlin et al. (2018) have proposed a deep learning method called Bidirectional Encoder Representations from Transformers (BERT) based on overcoming the limitations of *next* and *previous* token prediction procedures benefiting from Masked Language Modeling (MLM) (Taylor, 1953) by masking 15% of the sentence tokens fed into the architecture alongside the transformer encoder (Vaswani et al., 2017). Devlin et al. (2018) have extended their work by applying the same architecture in a Wikipedia corpora of 104 different languages, requiring not a single alignment signal and realising, if not outperforming, state-of-the-art score in many NLP tasks such as Part Of Speech Tagging and Named Entity Recognition. However, BERT demands significantly more machine effort (Wu and Dredze, 2019). Table 1 summarises the cross-language embedding models mentioned above according to the architecture and used corpus, the target languages and the evaluation methods.

3 Dataset Collection

3.1 Corpus Used

The main objective of this work is to provide an efficient Arabic-English cross-lingual word embedding models across different text domains. Indeed, we used a large dataset of parallel Arabic-English sentences mainly extracted from the Open Parallel Corpus Project[2] (OPUS) (Tiedemann, 2012). OPUS contains 90 languages, and more than 2.7 billion parallel sentences. This corpus consists of data from multiple domains and sources including: MultiUN Corpus (Daniel Tapias, 2010), OpenSubtitles (Creutz, 2018), Tanzil (Zarrabi-Zadeh, 2007), News-Commentary, United Nations (UN) (Ziemski et al., 2016), Wikipedia, TED 2013[3], GNOME[4], Tatoeba[5], Global Voices[6], KDE4[7] and Ubuntu[8] corpus. To train our models, we extract more than 93.9 million parallel sentences of Arabic-English from whole collection, this alignment contains more than 800 million Arabic tokens and 1 billion for English. More details about our dataset are given in Table 2.

3.2 Preprocessing and Normalization

Preprocessing is an important step in building any word embedding model as it can potentially significantly affect the end results. We first remove the punctuation marks, non letters, URLs, emojis and emoticons from the Arabic and English sentences. Additionally, we normalize Arabic sentences using the preprocessing suggested by Nagoudi et al.

[2] http://opus.nlpl.eu/
[3] http://www.casmacat.eu/corpus/ted2013.html
[4] https://l10n.gnome.org
[5] www.tatoeba.org
[6] https://globalvoices.org/
[7] http://i18n.kde.org
[8] https://translations.launchpad.net

CLWE Models	Corpus Used	Arch.	Languages	Evaluation
BiCVM (Hermann and Blunsom, 2014)	Europarl (Koehn, 2005), TED (Cettolo et al., 2012), RCV (Lewis et al., 2004)	CVM	English, German, French, Arabic, Spanish, Italian, Dutch, Brazilian	Cross-lingual classification
BiSKip (Luong et al., 2015)	UN corpus Koehn (2005)	Skip-Gram	English, German	Mono and bilingual word similarity, cross-lingual classification
BWESG (Vulić and Moens, 2015b)	UN corpus Koehn (2005)	Skip-Gram	English, Dutch	Mono and cross-lingual ad-hoc retrieval
BilBOWA (Gouws et al., 2015)	RCV (Lewis et al., 2004), WMT11 (2011)	CBOW	English, German, Spanish	Word translation, cross-lingual classification
ADAN (Chen et al., 2018)	UN corpus (Ziemski et al., 2016)	Skip-Gram	English, Arabic, Chinese	Domain Adaptation and Machine Translation
mBERT (Devlin et al., 2018)	Large Wikipedia Corpora	BERT	104 Languages (including Arabic)	POS Tagging and NER...etc

Table 1: Different cross-language word embedding models

(2017a):

1. The letters أ ، إ ، آ are replaced with ا while the letter ة is replaced with ه. Also, The letter ى followed by ء replaced with ئ.
2. We converted elongated words back to their original form, example : معاااااااهدة, which means *treaty* in English, and الجزائـــــــــر , which means *Algeria* will be converted to معاهدة, الجزائر.
3. In addition, we remove the stop-words from Arabic and English sentences.

4 Building ArbEngVec Models

4.1 Used Architectures

In Mikolov et al. (2013a) all the word embedding models (Collobert and Weston, 2008), (Turian et al., 2010), (Mnih and Hinton, 2009), (Mikolov et al., 2010), (Mikolov et al., 2013c) and (Mikolov et al., 2013b) have been compared and evaluated, and they show that CBOW (Mikolov et al., 2013c) and Skip-Gram (Mikolov et al., 2013b) models are significantly faster to train with better accuracy. Accordingly, we used the CBOW and Skip-Gram to build our Arabic-English cross-lingual word embedding models.

The CBOW (Mikolov et al., 2013c) and Skip-Gram (Mikolov et al., 2013b) are two shallow neural network architectures with a single hidden layer that learns similar vector representations for words with similar distributional properties. The CBOW model, predicts a targeted word w_t according to the context in which w_t appears by using a window of contextual words. While the Skip-Gram model, predicts the words around the word w_t (Mikolov et al., 2013a), as illustrated in figure 1.

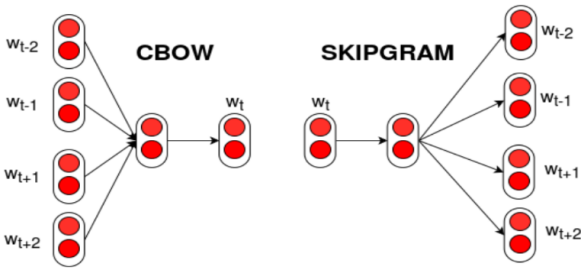

Figure 1: Architecture of CBOW and Skip-gram as described in (Mikolov et al., 2013b)

4.2 Proposed Models

In this section, we present our proposed ArbEngVec models. In order to learn our models, we have relied basically on shuffling the cor-

Corpus	Content	Documents	Sentences	Ar Words	En Words
MultiUN Corpus.	The official documents of the United Nations (UN)	67617	10.6M	263.1M	289.6M
OpenSubtitles.	A new collection of translated movie subtitles	104325	81.4M	501.5M	695.9M
Tanzil.	A collection of Quran translations	30 Quran Party	0.2M	7.9M	5.6M
News-Comment	A parallel corpus of News Commentaries provided by WMT for training Statistical Machine Translation (SMT)	7185	0.6M	15.4M	15.5M
UN.	A collection of translated documents from the United Nations originally	1	74.1k	3.3M	3.7M
Wikipedia.	A corpus of parallel sentences extracted from Wikipedia	1	0.2M	3.2M	3.5M
TED2013.	A parallel corpus of TED talk subtitles provided by CASMACAT	1	0.2M	2.4M	3.0M
GNOME.	A parallel corpus of GNOME localization files	1313	0.5M	2.4M	2.6M
Tatoeba.	A collection of translated sentences from Tatoeba	1	13.0k	90.1k	3.6M
GlobalVoices.	A parallel corpus of news stories from the web site Global Voices	7017	93.9k	2.1M	3.0M
KDE4.	A parallel corpus of KDE4 system messages	784	0.1M	0.7M	0.8M
Ubuntu.	A parallel corpus of the Ubuntu Dialogue Corpus	299	56.3M	0.2M	0.5M
EUbookshop.	Corpus of documents from the EU bookshop an online service and archive of publications from various European institutions	30	1.7k	80.0k	0.4M
Total	All the corpus used and extracted from OPUS	188606	93.9M	802.3M	1.0G

Table 2: Some statistics about the used dataset (Tiedemann, 2012)

	CBOW					Skip-Gram				
#Modes	**Top1**	**Top2**	**Top3**	**Top5**	**Top10**	**Top1**	**Top2**	**Top3**	**Top5**	**Top10**
Parallel	0.1%	0.5%	0.7%	1.2%	2.1%	2.8%	4.5%	6.1%	6.1%	9.3%
W. by W.	4.1%	11.3%	17.4%	25.3%	37.2%	60.6%	73.5%	78.3%	86.8%	92.4%
Random	57.7%	71.4%	79.2%	85.3%	90.5%	62.4%	74.2%	78.4%	87.5%	**93.8%**

Table 3: Intrinsic evaluation results of ArbEngVec models

pus as in Vulić and Moens (2015b), with one major difference choosing sentence-aligned parallel data rather than their comparable document-aligned choice. Indeed, we propose to use three methods for learning our models: *Parallel Mode*, *Word by Word Alignment Mode* and *Random Shuffling Mode*.

4.2.1 Parallel Mode

To make clear that shuffling methods adds cross-lingual improvements, we decided to train a model without any alignment. For example, let S_{ar} and S_{en} be Arabic and English sentences:

S_{ar} = "الولدان الصغيران شقيقان".

S_{en} = " *The young boys are brothers*".

The pair (S_{ar}, S_{en}) were fed directly to the training as follows: "*young, boys, brothers,* الولدان, الصغيران, شقيقان".

4.2.2 Word by Word Alignment Mode

The second method used on the same corpus type with aligning pairs *word by word* and paying attention to sentences length and start aligning with the longest (the short sentence words will be surrounded with those of the long sentence). This method supports using pairs with almost equal lengths. In this situation, stop-words removal preprocessing step is highly blessed. We shall continue with the sentences of the previous example, the input of the training is : " *young,* الولدان, *boys,* الصغيران, *brothers,* شقيقان".

4.2.3 Random Shuffling Mode

In this method, we put each pair of bilingual sentences as a list that contains their words and shuffle it **randomly** and separately from the rest of the corpus to have a list of combined English-Arabic tokens. As shown in our example : " *young,* الصغيران, الولدان, *boys, brothers,* شقيقان".

4.3 Parameters and Training Environment

Training word embedding models require the choice of some parameters affecting the resulting vectors. For our CBOW models we have used recommended parameters values proposed by (Mikolov et al., 2013c). Thus, we set the $vector\ size$ to 300, the $window$ = 5, and $Frequency\ threshold$ = 100. Regarding the Skip-gram models we have chosen Negative Sampling with $negative$ = 5 instead of Hierarchical Softmax. Worth mentioning that all models were trained on 10 epochs with Řehřek and Sojka (2011) GenSim tool.

Concerning the training environment, we have used *Google Colaboratory*[9] research project (also known as *Colab*) for training our model variants. It is a perfectly prepared developing environment with no requirements but a browser. This environment provides a free 12 GB of GPU, also access to *Google Drive* personal account for saving and loading files and there are many other services that can be plugged into it.

5 Evaluation

Usually multilingual models go against two aspects of evaluation methodology: maintain monolingual aspect and provide the other cross-lingual. Clearly for us, after creding on the shuffle we lost the former willingly to stick around the latter. Preserving the model's monolingual behaviour requires keeping words in a semantic meaningful order, which is exactly what happens with our first parallel (non-shuffling) model with completely skewed cross-lingual aspect. To clarify that, we have evaluated our models through Semantic Textual Similarity as extrinsic, and Word Translation as intrinsic.

5.1 Intrinsic Evaluation

In this step, we basically focused on word translation following (Gouws et al., 2015) evaluation procedure, so we generated a 1000 tuples starting with choosing random 1000 words from the model vocabulary. Then, we find their k-$closest$ (k most similar) cross-lingual words based on the cosine similarity in our six ArbEngVec models. In fact, we have used five different values of k to generate the 1-$closest$, 2-$closest$, 3-$closest$, 5-$closest$ and 10-$closest$ words.

[9]https://colab.research.google.com/

For example, Table 4 shows the 5-$closest$ words of مالزيا and $weapons$ in our *random Skip-Gram* model. Afterwards, we calculate the accuracy of each range, which has been calculated by giving a value 1 to each word couple that represents a translation, we make sure that the word provided by our model is a translation with comparing it to Google Translate API's bag of words, if this comparison comes negative we compare manually, if also manual comparison comes negative we give negative score 0. Eventually we count the average of the 1000 scores. Results of the six studied models are provided in Table 3.

Discussion. Parallel results were so dim bilingually as Table 3 shows, but monolingual aspect was preserved especially in CBOW variant. This fact is illustrated in Table 5, the same 5-$closest$ words of ماليزيا and $weapon$ using Parallel CBOW model. Switching to *word by word* alignment method, both variants gave promising results and notably Skip-gram's by an average of 59.26% from CBOW, and these are a consequence of getting word translation pairs at the context window range but still since Arabic and English are structurally different this alignment method had its inconvenience. Arriving to *random shuffle* variants which have given the best results and again Skip-Gram with average of 2.44% better than CBOW.

5-closest (مالزيا)	5-closest (weapons)
malaysia, قرغيزيستان, منغوريا, تونغا, كودت	الأسلحة, الدمار, أسلحة, mass, indiscriminite

Table 4: A sample of 5-$closest$ words of مالزيا and $weapons$ in our Random Skip-Gram model

5-closest (مالزيا)	5-closest (weapons)
مدغشقر, المكسيك, ليسوتو, نيجريا, نيبال	arms, weaponry, warheads, missiles, arsenals

Table 5: A sample of 5-$closest$ words of مالزيا and $weapons$ in our Parallel CBOW model

5.2 Extrinsic Evaluation

Extrinsic evaluating means surveilling the model performance under real-world Natural Language Processing tasks use. Our choice fell on Semantic Sentences Similarity (STS) task. To estimate the semantic similarity between the Arabic-English sentences, we have used the WE-based approach proposed by Nagoudi et al. (2017b) jointly with our ArbEngVec models. In fact, we have had STS2017-Eval[10] datasets drawn from the shared taskSemEval-2017 Task1: STS Cross-lingual Arabic-English (Cer et al., 2017). The sentence pairs of STS2017-Eval have been manually labelled by five annotators, and the similarity score is the average of the annotators judgments. Afterwards, in order to evaluate the performance of each model, we calculate Pearson correlation between our assigned semantic similarity scores and human judgement. Table 6 reports the results of the six studied models.

# Modes	**CBOW**	**Skip-Gram**
Parallel.	6.3%	18.1%
W. by W.	49.4%	73.6%
Random.	52.8%	**75.7%**

Table 6: Extrinsic evaluation results of ArbEngVec models

Discussion. These results indicate that when the *parallel* alignment is used the correlation rate gets very low in both architectures. This is due to the distance of every word and its translation in the parallel sentences pair shape. However, when applying the *word by word* alignment the correlation rate is clearly outperformed to 49.4% and 73.6% with the CBOW and Skip-Gram model respectively. Additionally, the observed results indicate that the *random shuffling* method with Skip-Gram model is the best performing method with a correlation rate of 75.7%.

5.3 Models Visualization

As part of the discussion, we have chosen to illustrate our models using *pyplot* scatters with Maaten and Hinton (2008) *t-SNE* algorithm. We provide these visualizations by choosing 20 arbitrary words from our vocabulary, run 4-*closest* similarity to each word and finally project all of them on the 2-dimensional plot. Starting with *parallel* mode models, charts show that distance between Arabic markers are distant from others of English comparing to those of the same language. Same thing can be said on the situation that concerns *word by word* method CBOW variant with less distant languages but still marker bags most often do not include translation pairs. Eventually, *random* variant charts make it clear that close markers include translation pairs alongside mono and cross-lingual similarities, six model charts are in figure 2. Especially for Skip-Gram variant, supposedly that t-SNE feature reduction procedure got rid of both language characteristics, as figure 3 shows, words and their translations most often appear next to each other.

6 Conclusion

In this paper, we have presented the open source project named ArbEngVec. This project provides several Arabic-English cross-lingual word embedding models. The embedding models are learned through a large dataset of parallel Arabic-English sentences. Additionally, we evaluated the ArbEngVec models via extrinsic and intrinsic evaluations. In the extrinsic evaluation, we used the cross-language semantic similarity task to test the capability of our models to capture the semantic and syntactic properties of words in two different languages. While in the intrinsic evaluations, we employed the embedding vectors to evaluate the word translation task.

As future work, we are going to use these models with those of other classical NLP techniques, including word sense disambiguation, named entity recognition to make more improvement in the Arabic-English cross-language semantic similarity and plagiarism detection. We also are going to aim on finding better word alignment methods to improve features capturing regarding the transfer between Semitic and Germanic languages.

[10] http://alt.qcri.org/semeval2017/task1/index.php?id=data-and-toolsb

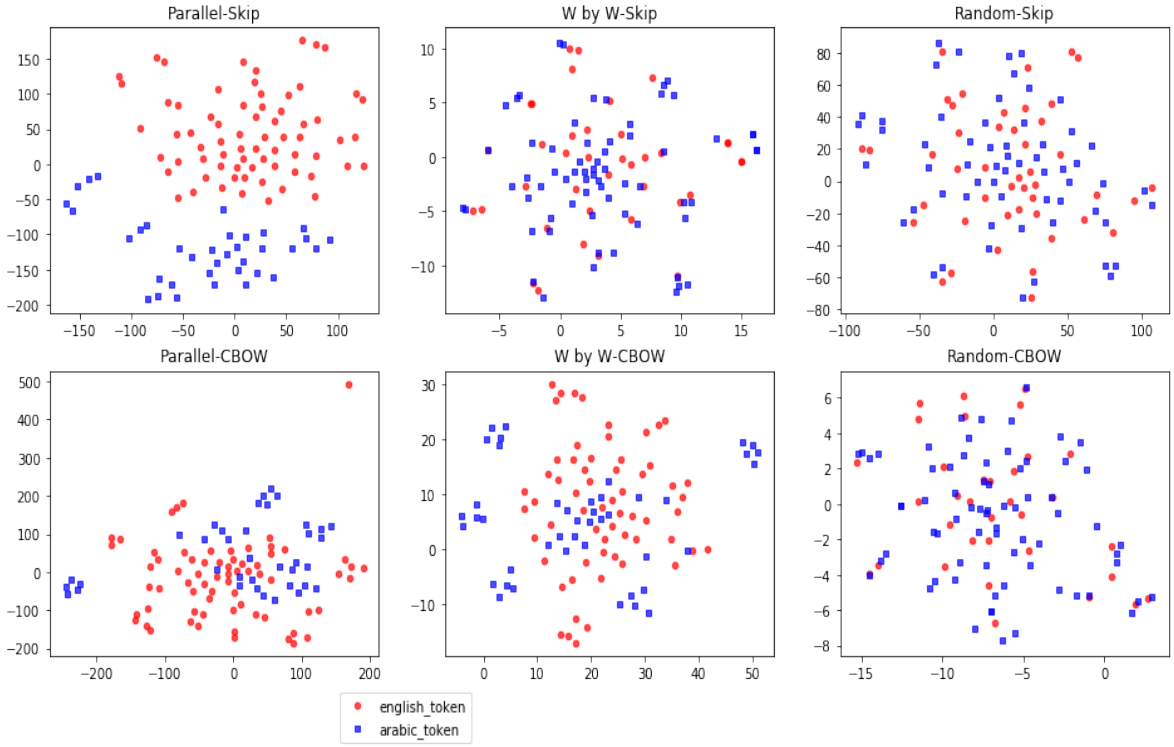

Figure 2: Charts of the model's six variants

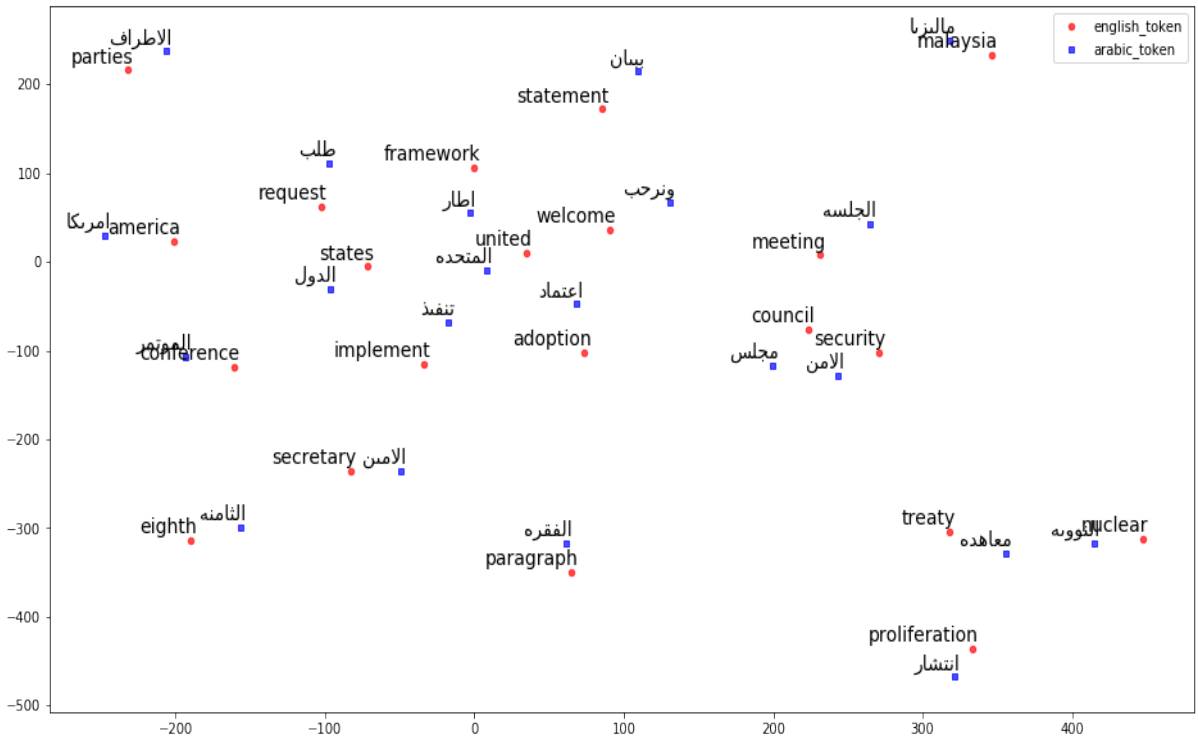

Figure 3: Chart of Random Skip-Gram model

References

Duygu Ataman, Jose GC De Souza, Marco Turchi, and Matteo Negri. 2016. Fbk hlt-mt at semeval-2016 task 1: Cross-lingual semantic similarity measurement using quality estimation features and compositional bilingual word embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 570–576.

Alberto Barrón-Cedeño, Parth Gupta, and Paolo Rosso. 2013. Methods for cross-language plagiarism detection. *Knowledge-Based Systems*, 50:211–217.

Phil Blunsom and Karl Moritz Hermann. 2014. Multilingual models for compositional distributional semantics.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. ACL.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit3: Web inventory of transcribed and translated talks. In *Conference of European Association for Machine Translation*, pages 261–268.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

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

Mathias Creutz. 2018. Open subtitles paraphrase corpus for six languages. *arXiv preprint arXiv:1809.06142*.

Stelios Piperidis Jan Odjik Joseph Mariani Bente Maegaard Khalid Choukri Nicoletta Calzolari Daniel Tapias, Mike Rosner. 2010. Multiun: A multilingual corpus from united nation documents.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yerai Doval, Jose Camacho-Collados, Luis Espinosa-Anke, and Steven Schockaert. 2018. Improving cross-lingual word embeddings by meeting in the middle. *arXiv preprint arXiv:1808.08780*.

Jérémy Ferrero, Frédéric Agnes, Laurent Besacier, and Didier Schwab. 2017. Using word embedding for cross-language plagiarism detection. *arXiv preprint arXiv:1702.03082*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. *arXiv preprint arXiv:1404.4641*.

Tom Kenter and Maarten De Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1411–1420. ACM.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Bilingual word representations with monolingual quality in mind. pages 151–159.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *In: ICLR: Proceeding of the International Conference on Learning Representations Workshop Track*, pages 1301–3781.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Hlt-naacl*, volume 13, pages 746–751.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. Curran Associates, Inc.

El Moatez Billah Nagoudi. 2018. Arb-sen at semeval-2018 task1: A new set of features for enhancing the sentiment intensity prediction in arabic tweets. In *SemEval@ NAACL-HLT*, pages 364–368.

El Moatez Billah Nagoudi, Jérémy Ferrero, and Didier Schwab. 2017a. Lim-lig at semeval-2017 task1: Enhancing the semantic similarity for arabic sentences with vectors weighting. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 134–138.

El Moatez Billah Nagoudi, Jérémy Ferrero, Didier Schwab, Hadda Cherroun, et al. 2017b. Word embedding-based approaches for measuring semantic similarity of arabic-english sentences. In *International Conference on Arabic Language Processing*, pages 19–33. Springer.

El Moatez Billah Nagoudi, Ahmed Khorsi, Hadda Cherroun, and Didier Schwab. 2018. A two-level plagiarism detection system for arabic documents. *Cybernetics and Information Technologies*, 20.

El Moatez Billah Nagoudi and Didier Schwab. 2017. Semantic similarity of arabic sentences with word embeddings. In *Third Arabic Natural Language Processing Workshop*, pages 18–24.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Radim Řehřek and Petr Sojka. 2011. Gensimstatistical semantics in python. *statistical semantics; gensim; Python; LDA; SVD*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1555–1565.

Wilson L Taylor. 1953. cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. 2012:2214–2218.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ivan Vulić and Marie-Francine Moens. 2015a. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 363–372. ACM.

Ivan Vulić and Marie-Francine Moens. 2015b. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. pages 363–372.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. *arXiv preprint arXiv:1904.09077*.

Francisco Zamora-Martinez and Maria Jose Castro-Bleda. 2011. Ceu-upv english-spanish system for wmt11. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 490–495. Association for Computational Linguistics.

Hamid Zarrabi-Zadeh. 2007. Tanzil project. *URL: http://tanzil. net/wiki/Tanzil_Project*.

Dong Zhou, Mark Truran, Tim Brailsford, Vincent Wade, and Helen Ashman. 2012. Translation techniques in cross-language information retrieval. *ACM Computing Surveys (CSUR)*, 45(1):1.

Michal Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The united nations parallel corpus v1. 0. In *Lrec*.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.