Compositional Representations of Language Structures in MultiLingual Joint-Vector Space

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

Şaban DALAMAN

in partial fulfillment for the
degree of Master of Science

in
Data Science

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Data Science.

APPROVED BY:

Asst. Prof. Barış Arslan .....
(Thesis Advisor)

Assoc. Prof. Ahmet Bulut .....

Prof. Dr. Selim Akyokuş .....

This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences of İstanbul Şehir University:

DATE OF APPROVAL: 08.05.2018

SEAL/SIGNATURE:

Declaration of Authorship

I, Şaban DALAMAN, declare that this thesis titled, 'Compositional Representations of Language Structures in MultiLingual Joint-Vector Space' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 08.08.2018

"A joke for semanticists."
Q: What is the meaning of life?
A: life

Compositional Representations of Language Structures in MultiLingual Joint-Vector Space

Şaban Dalaman

Abstract

After the recent developments in Artificial Neural Networks and deep learning techniques, representation learning has become the focus of many research interests. In the field of Natural Language Processing, representation learning techniques have gained many implementation advances and improved different tasks compared to any other method. One of the primary research topics in this area is to construct compositional representations of discrete language structures in multilingual joint-vector space. In this thesis study, several techniques from deep learning and NLP are combined to investigate their potential impact on NLP tasks.

For this purpose, four different composition vector models (CVM) by using tokens and morphemes as basic language structures are studied. To construct tokens and morphemes, first, a parallel corpus is segmented into discrete objects via tokenization and morphological analysis. Several hierarchical composition methods via bilingual method are employed to construct the embeddings of these structures. Bilingual models are trained by using sentence-aligned corpora for four languages. The models learn how to employ compositional vector models and construct embeddings of sentence constituents as well.

Two different test scenarios are performed to evaluate different CVMs. The first one is paraphrase test. In this case, the bilingual models using CVMs are trained with each language pair L1-L2 (English, Turkish, German and French) parallel corpus. Then the models are tested by evaluating their performance in finding the corresponding pairs correctly from 100 randomly selected sentences from each L1-L2 pair.

The other test scenario is cross-lingual document classification. In this case, the trained models are employed by a document classifier model to evaluate their performance in classification task by first training in L1 documents and then testing with L2 documents.

Keywords: Natural Language Processing, Language Modelling, Deep Learning, Neural Networks, Multilingual Word Embeddings, Distributed Representations, Representation Learning

Çok Dilli Eklem-Vektör Uzayda Dil Yapılarının Bileşim Temsili

Şaban Dalaman

Öz

Son dönemdeki yapay sinir ağları ve derin öğrenme tekniklerinde ki gelişmelerle beraber, temsili öğrenme pek çok araştırmanın odak noktasında yer almaya başladı. Doğal dil işleme(DDİ) alanında, temsili öğrenme tekniklerinin uygulamasında ve diğer metodlara göre DDİ problemlerinin çözümünde ilerleme sağlamıştır. Bu alandaki ana araıstırma konularından biri, dil yapılarının ortak çok dilli uzayda birleşimsel temsillerini oluşturmaktır. Bu çalışmanın hedefi derin öğrenme ve DDİ mede kullanılan bazıtekniklerin birleştirilerek temsillerin DDİ uygulamalarındaki etkisini araştırmaktır.

Bu amaçla 4 değişik birleşim vektör modeli üzerinde çalışılmıştır. Token yada morpheme gibi dil yapılarının temsil uzaylarının oluşturulması için ilk olarak tokenizasyon yada morfolojik ayrıştırma ile paralel korpus hazırlanmış sonra değişik hiyerarşik birleşim metodları ikili-dil modelleri üzerinden kullanılmıştır. İkili-dil modelleri, 4 dil için hazırlanan cümle sıralı korpuslar kullanılarak eğitilmiştir. Bu sayede, model birleşimsel vectör modelini kullanarak cümle elemanlarının temsillerini oluşturmayı öğrenmektedir.

Değişik birleşimsel vektör metodlarını değerlendirmek için iki test senaryosu kullanılmıştır. İlki açımlama testidir. Bu senaryoda ikili model, birleşimsel vektör modelini kullanarak eğitilir. Sonra paralel korpusdan iki dil için seçilen karşılıklı cümle çiftlerinin karşılaştır ılmaları ile performansları hesaplanır.

Diğer test senaryosu ise gözetimli döküman sınıflama testidir. Bir dilden seçilen dökümanlar kullanılarak eğitilen sınıflandırıcı , diğer bir dilden seçilen test dökümanları ile test edilir. Dökümanlar değişik konu başlıkları için pozitif ve negatif olarak işaretlenmiştir. Sınıflandırıcı pozitif ve negatif örnekleri ayırmayı öğrenmektedir.

Anahtar Sözcükler: Doğal Dil İşleme, Dil Modelleme, Yapay Sinir Ağları, Derin Öğrenme, Gösterimsel Öğrenme

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisors Prof. Onur Güzey and Prof. Barış Arslan for their continuous support during my study at Şehir University. I thank them for their patience, motivation, and knowledge. With their guidance, I was able to complete my thesis. They have been great mentors as it should be! I would also like to thank Prof. Ahmet Bulut, Prof. Ali Çakmak and Prof. Mehmet Baysan for being my instructors during the master program, helping me to learn the subjects and complete it successfully.

I would also like to thank the rest of my thesis committee: not only for their insightful comments and encouragement but also for the hard questions which incented me to widen my research from various perspectives.

I also thank my fellow lab mates for the stimulating discussions, help and for being a great company in the lab.

Contents

List of Figures

List of Tables

Abbreviations

LSTM	**L**ong **S**hort **T**erm **M**emory
Bi-LSTM	**B**i-directional **L**ong **S**hort **T**erm **M**emory
NLP	**N**atural **L**anguage **P**rocessing
DDİ	**D**oğal **D**il **İ**şleme
CVM	**C**ompositional **V**ector **M**odels
CLDC	**C**ross **L**ingual **D**ocument **C**lassification

Chapter 1

Background

1.1 Introduction

Distributed representations have become the basis for many tasks in NLP. Word representations provide richer representations than traditional methods. It has been shown that they capture both syntactic and semantic information of a language. Language modeling in Bengio et al. [1] and dialog analysis in Blunsom and Kalchbrenner [2] are among the successful applications of distributed representations.

This approach can be extended to joint-space embeddings for multilingual data. The first example of this approach was given by Hermann and Blunsom [3]. In their approach, a novel unsupervised technique has been proposed. The model first learns semantic representations using parallel corpora, and employs these representations through composition models. Instead of learning a single language word representations, in their approach, the model learns to represent common syntactic and semantic structures across multilingual joint-representation space.

The primary purpose of this study is to investigate the effect of composition models combined with basic language structures such as words, tokens or morphemes on computing multilingual embeddings. We present experiment results on parallel corpora for 4 languages: English, French, German, and Turkish. The model first learns multilingual representations among languages. Then we employ the learned representation to a text classification problem and present the results. We apply two different language processing steps during the learning process of multilingual representations; tokenization

and morphological analysis. Two versions of parallel corpora are prepared. In the first version, training corpus is segmented into sentences and tokens, and in the second version corpus is processed with a morphological analyzer and segmented into morphemes. Morfessor by Creutz and Lagus is used to create morphological analysis models for each language [4, 5]. With these analysis models, morphological segmentation of words for each language is found, and morpheme corpora are prepared. Finally, we prepare semantic representation space for each type of parallel corpora. Then we compare the classification results with the previously published experiments and our results in the learning representation with different basic language structures.

1.2 Related Works

Recently, cross-lingual representation has been quite popular. In the literature, one may find several main approaches for cross-lingual embeddings. They can be grouped as monolingual mapping, cross-lingual training, and joint-optimization by Ruder [6]. Different types of data representation techniques are used for different purposes. They can be word-aligned, sentence-aligned or document-aligned corpora. Moreover, depending on the target task, the corpus can be prepared by bilingual lexicon, for example, for translation. Our study concentrates mainly on cross-lingual training. In this approach, training process depends exclusively on optimizing the objective function for cross-lingual embedding.

The first example was given by Hermann and Blunsom [7]. The authors use a parallel sentence-aligned corpus. They train two models in parallel to compute sentence representations. Using the objective function the distance between sentences from two languages is minimized to reach sentence embedding. In this thesis study, we are going to use and extend by Hermann and Blunsom approach by adding new composition vector models [7]. Another approach comes from Larochelle et al. [8]. They use a monolingual tree-based auto-encoder to encode the input sentence and another encoder to decode to form target sentence. Kocisky et al. has suggested an approach to learn word embeddings and alignments at once such that a word in the source sentence is used to predict the word in the target sentence [9]. Hermann and Blunsom extended their approach in [3] to cover document embeddings [7]. In their application, first sentence embeddings are computed then recursively the same composition function applied to compute document

embeddings. Chandar et al. [10] modified the approach proposed by Larochelle et al. [8]. Instead of using a tree-based decoder, they suggested a bag of words technique using a sparse binary vector of word occurrences. Similar to sentence-aligned methods, Manning, Pham and Luong proposed a method to learn sentence representations by extending paragraph vectors Mikolov et al. [11] to the multilingual setting [12]. Sentences of different languages are aligned to share the same vector. In her thesis [13], Sohsah applied two simple composition function mentioned by Hermann and Blunsom using corpora prepared by morphological analysis [7]. It was the first attempt to use morphemes in Turkish instead of tokens. Turkish is a morphologically rich language, and easily a number of words can blow up in the corpus without carrying crucial semantic information. So instead of word or token, root morpheme can be more efficient to build up embeddings in Turkish language.

Chapter 2

Representation Learning

2.1 Distributed Representation

Distributed representations are the continuous representations of discrete objects (token or morpheme) in a language. Usually, word representations are found for a language and used for other NLP tasks such as classification, sentiment analysis etc. Here we concentrate on finding sentence representations from multilingual parallel corpora and search for how the usage of joint-space representations effect the performances of other NLP tasks. Our model learns to represent language structures like word, morpheme or sentence via a composition function such that distributed representations in joint-space help a model to learn a shared meaning between similar language constituents. Besides, these representations can be used to capture the syntactic and semantic content of the documents.

The model uses a multilingual objective function, and as a part of it, a composition vector model (CVM) is used to compute semantic representations of sentences. CVM computes and learns the representation in a hierarchical manner, first token or morpheme and then sentence representation.

CVM models use training signals like parse tree or annotated data additionally. In our model, CVM uses a sentence-aligned parallel corpora. By using the same objective function and CVM among all languages, we can learn a joint-semantic representation. Moreover, CVM models learn how to compute sentence level representation by using a

token or morpheme representations. Common semantic representations can be found by employing parallel corpora of multiple languages.

Distributed representation can be learned mainly by using two approaches. The first one is topic modeling techniques. Window method is used to compute embedding. From a large corpus, the word occurrence frequency is computed within a certain window size of the target word. These word counts are subsequently used to compute embeddings. LDA and LSA techniques use this method. The main drawback of these techniques is that they use document level context and try to capture the topics from whether some words are used or not. They do not use the syntactic content.

Natural language models have recently become more preferred models for computing distributed representations. By the work of Collobert and Weston [14], Mnih and Hinton [15], Mikolov et al. [16], it has been achieved great benefit in language modeling. Neural network architectures have been used broadly for learning word embeddings and been shown that they could improve supervised NLP tasks. Unsupervised word embeddings can easily be used with different NLP tasks and improve their performances.

Most of the research on distributed representations is concentrated on a single language. English as a language with a large amount of resources has been the main focus. However, much work can be done by transferring linguistic information among languages by learning multilingual embeddings. A feasible approach is to learn joint embedding from shared semantic space across multiple languages.

2.2 Compositional Distributed Semantics

Through representing individual language units such as words or tokens, a distributional representation encodes the necessary information about the corresponding word or token. Such information can be encoded by using collocational methods in Collobert and Weston [14] and Erk et al. [17].

Individual word representations, however, can not be sufficient. A semantic representation of a larger structure such as sentence or phrase is better to encode the meaning for a number of reasons. However, the same method for learning word level representation

may not be so useful for larger structures. Instead, it is better to focus on learning composition functions for large structures and use them to derive the representation of its parts. In the literature, one may find from simple composition function like using bi-grams in Mitchell and Lapata [18] or some mathematical operations like multiplication of representation of constituents to more complex ones like a matrix or tensor composition or convolutional networks and LSTMs.

Mitchell and Lapata in list some examples of simple composition functions applied to bi-grams [18]. Word level representations can be extracted by recursively employing these functions. Additionally, Clark and Pulman has suggested a tensor-based model as a semantic composition [19]. Recently, various forms of RNNs have successfully been used as semantic composition function and applied to the related NLP tasks. Such models include recursive auto-encoders by Socher and Manning et al. [20], matrix-vector recursive neural networks by Socher and Huval et el.[21], untied recursive neural networks Hermann and Blunsom [22] or convolutional networks by Hermann and Blunsom [23].

2.3 Vector Space Models

A vector space is an algebraic model of any object in machine learning in general. Specifically, in NLP it is a model for the representation of language structures. These structures can be from lowest level such as a single character or character n-grams to morphemes, tokens, sentences, paragraphs up to any size of documents. It is usually vector space with a predefined dimension. There are a matrix or tensor variants of it. Once constructed by different methods, they can be used for many NLP tasks. Using similarity metrics, for example, word, sentence or documents similarity can be studied.

A vector model for a language element is called embedding in the corresponding vector space. Embedding is usually represented as a vector in n-dimensional vector space. In NLP, embeddings for language structures such as word, token or morpheme are extensively used. One of the methods used to represent language units as vectors is one-hot encoding. Its dimension is the number of distinct language units in the corpus. Each vector that represents a unit consists of all elements as zero except the one that indicates

that feature. That vector element is assigned as one. However, this kind of representation is not appropriate to represent syntactic and semantic properties of language structures. Instead, a distribution of values for high dimensional vector can be used. Each component of a vector is a real value. Such representations can model language structures more compactly. Instead of vectors space with huge number of dimension as in the case of one-hot encoding, a low-dimensional vector is enough to provide the information to represent the language features. It can also convey syntactic and semantic properties of language constituent.

Chapter 3

Natural Language Processing

3.1 Terminology

Natural language processing is an inter-disciplinary field that combines computational linguistics and machine learning or artificial intelligence. Mainly, it is interested in human languages and text or documents in human languages. Several essential NLP tasks should be explained to make our thesis study understandable.

Stemming is the process of removing affixes (suffixed, prefixes, infixes, circumfixes) from a word and obtaining a word stem. For example :

running $\rightarrow$ *run*

Lemmatization is a process of capturing canonical forms based on a word's lemma. For example, stemming the word "better" would not find its citation form. However, lemmatization would result in the following:

better $\rightarrow$ *good*

Corpus indicates a collection of documents. Usually, it is formed from a single language of texts. In our case, we use documents from multiple languages – called multilingual corpora (plural of corpus).

3.2 Token And Tokenization

A typical NLP task starts with segmenting corpus into words, sentences, phrases, and symbols. These language constructs are linguistically significant. Segmenting corpus into indicative language units is crucial for learning effective representations. Tokenization is one of these processes. The tokens may be words or numbers or the punctuation mark. Tokenization is done by locating word boundaries and splitting sentences into words. The ending point of a word and the beginning of the next word are called word boundaries.

3.3 Morpheme And Morphological Analysis

As indicated before, tokenization is a kind of segmentation of corpus into words. However, even words can be segmented into meaningful smaller structures. These are called *morphemes*. Morpheme is defined as the "minimal unit of meaning". Morphology is the study of structure and formation of words from morphemes in a language. Most morphemes are called affix. There are two kinds of affix, a prefix, and a suffix. Languages can be classified according to their morphological structures, and some of them have rich morphological structure. Depending on the language studied, morphemes can be more useful in a representation learning task in NLP. English language, from the morphological point-of-view, can be considered a straightforward language. However, the other languages may behave rather in different ways. In *Agglutinative languages*, simple words are usually combined without change of form to express compound ideas. For example, Turkish and Finnish are such languages. In Finnish, a single verb may appear in thousands of different forms Creutz and Lagus [5]. While working on learning representation of language structures, usually word is taken as the smallest meaningful unit, and its embedding is computed. However, taking language morphological structures into account, learning representation can be started from morphemes then ascended up to words and sentences. The smaller units can be combined to make larger structures.

Chapter 4

Problem Definition

One of the significant problems of working in NLP problems is finding a corpus large enough to model language structures. Because of this limitation, most of the work has been done on the high-resources languages like English. Distributed representation is the most accepted way to encode relations between *words*. Recent studies have shown compositional semantic representations can be applied to monolingual applications. Moreover, some initial work has been conducted successfully on learning representations across languages.

One of the solutions to data scarcity problem can be solved by learning multilingual representations using parallel corpus between a high-resource language and a low-resource language. Besides, a multilingual representation can capture semantic relations between languages. In this thesis, we have studied possible solutions to data scarcity problem by employing compositional vector models to learn embeddings of aligned sentences and applying them to a cross-lingual document classification problem where training document resources of one language are ubiquitous but very scarce for another language.

To tackle the problem of finding the correct compositional vector model to compute sentence embeddings from its constituents, we have studied two methods for segmenting sentences and words. Firstly, sentences were segmented into tokens. So the basic language unit was token. As a second step, we have applied morphological segmentations to words as well as tokenization. So the basic language unit was *morpheme*. Then bilingual models using different CVMs were trained by using the parallel corpus segmented

either by tokenization or morphological analysis. There are two outputs of the training process of a bilingual model. One of them is embedding of basic language units for each language, and the other is CVM model parameters to compute sentence embedding from these basic unit embeddings.

In summary, while studying the problem of extracting distributed representations from a language corpora, usually two main difficulties are tackled: data scarcity problem for some languages and the fact that languages are not equal regarding grammar and semantics. It is therefore inappropriate to treat all languages in the same manner. The central claim of this study is to find an adequate joint-space embedding of common structures for languages in a hierarchical manner and explore their impact on common NLP tasks - in this case, document classification.

Chapter 5

Methodology

5.1 Tokenization And Morphological Analysis

In our study, for tokenization, we use The Natural Language Toolkit (NLTK). NLTK is a collection of open source programs including corpus readers, tokenizers, stemmers, taggers, parsers, etc. It also provides sample programs written in Python.

Morphology is the study of structure and formation of words in a language. Languages can be classified according to their morphological structures, and some of them have rich morphological structure. Depending on the language studied, morphemes can be more useful in a representation learning task in NLP. In our study, we investigated also morpheme representations as well as token representations and their effect on NLP tasks.

Constructing a morphological analyzer based on linguistic rules is considerably complicated and requires too much time. Unsupervised morphological analyzers have become popular because of its unsupervised nature and dependency on only data, not grammatical rules. Therefore to segment corpus into morphemes, we decided to use an unsupervised morphological tool, Morfessor, Creutz and Lagus [4] [1]. Morfessor is a family of methods for unsupervised morphological segmentation. Morfessor first separates words recursively to achieve an objective called Minimum Description Length. Then it labels morphemes with affix type as prefix, stem or suffix. Finally, it learns how to form a word from affixes.

[1] http://morpho.aalto.fi/projects/morpho/

Morphessor models were trained by using Wikipedia datasets [2] for each language set: English, Turkish, German and French. By using these models, corpus for each language and cross-lingual documents are processed and segmented into morphemes.

5.1.1 Out of Vocabulary Issue - OOV

During training of the models to compute distributed representations, there is an important issue. It is called *out of vocabulary* case. During the testing phase, the model may be in a position to handle some language units (words, tokens or morphemes) whose representation it does not know. In NLP tasks, this usually happens because it is impossible that training set covers all language features. A common approach for solving this issue is to replace some corpus constituents with UNK symbol either randomly or by selecting the constituents that have occurrence frequency below the certain limit in the corpus. By replacing some elements with UNK symbol before training, the model can learn its representation, and during testing whenever it encounters an unknown language unit, the representation of UNK symbol can be used instead of the unknown element.

5.2 Compositional Vector Models

Prior work on learning compositional vector representation by Herman and Blunsom [3]. It removes the requirements for parse trees or annotated data limitations. Their idea is that given enough parallel data finding a shared representation of parallel sentences would capture the common syntactic and semantic structures between these sentences. Since languages have different semantic structures, we compute deeper semantic representations. First, we define an energy function as follows: two functions are defined

$$f : x \rightarrow R^d \qquad \textit{and} \qquad g : y \rightarrow R^d \tag{5.1}$$

These functions map sentences from language x and y onto a semantic representation embedding space R^d . Given a parallel corpora C, an energy function is defined as:

[2] http://opus.nlpl.eu/Wikipedia.php

$$E(x, y) = \| f(x) - g(y) \|^2 \tag{5.2}$$

The objective is to minimize E for all aligned sentences in a given parallel corpus. By using the energy function, we can define an objective function:

$$J(\Theta) = \sum_{(x,y)\in C} (E(x, y) + \frac{\lambda}{2} \| \Theta \|^2) \tag{5.3}$$

Here Θ is the set of model parameters.

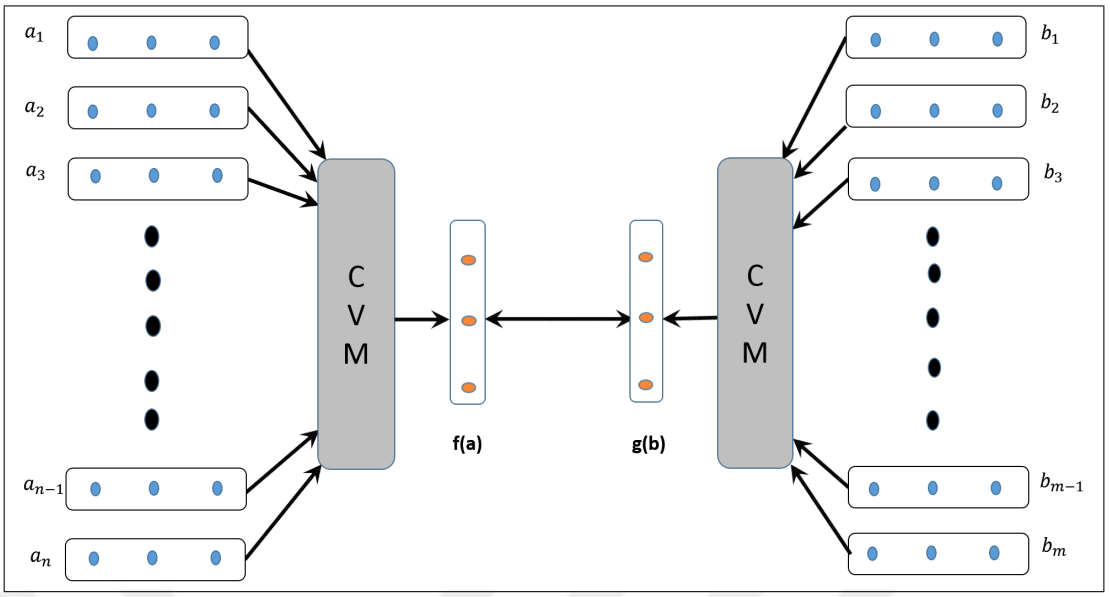

FIGURE 5.1: CVM Diagram

Figure 5.1 shows the diagram of a model for multilingual representations. It minimizes the distance between embedding vectors of sentences from two languages in joint-space. CVM is used to compute sentence representation from low-level components such as token or morpheme.

The objective function could use any composition function f and g. Since we intend to find a generic model that could be used for a wide range of languages, composition function used should not depend on any syntactic information. In our study, we evaluated four different CVM functions.

5.2.1 Additive

The first function is called additive model - ***Add***. It computes sentence vectors by simply adding language constituent representations.

$$f(x) = \sum_{i=1}^{n} x_i \tag{5.4}$$

Equation 5.4 is the formula of Additive CVM.

5.2.2 Bi-Tanh

The second function is called bi-gram model ***BI-Tanh***. This function computes sentence vectors by adding representations of sentence constituents (tokens or morphemes) as bi-grams and applying a nonlinear function Tanh to bi-grams and summing all.

Unlike Add model, BI model may capture nonlinear interactions between constituents.

$$f(x) = \sum_{i=1}^{n-1} \tanh(x_{i+1} + x_i) \tag{5.5}$$

Equation 5.5 is the formula of Bi-Tanh CVM.

5.2.3 LSTM

A Recurrent Neural Network (RNN) is a class of artificial neural networks (ANN) where there are connections between nodes such that it can represent the dynamic model of an ordered sequence or time-dependent data. This behavior makes them an appropriate type neural network model to be used in modeling of language structures. Different kind of neural network blocks can be used to build an RNN model. One of them is Long Short Term Memory (LSTM) unit.

A vanilla LSTM unit has three gates (input, forget and output), block input, a single cell, an output activation function, and peephole connections. The output of the block is recurrently connected back to the block input and all of the gates Schmidhuber et el. [24].

Here x^t is the input vector at time t, W are rectangular input weight matrices, R are square recurrent weight matrices, p are peephole weight vectors and b are bias vectors. Functions $\odot$, g and h are point-wise non-linear activation functions: logistic sigmoid is used as the activation function of the gates and hyperbolic tangent is usually used as the block input and output activation function. The point-wise multiplication of two vectors is denoted with $\odot$:

$$z^t = g(W_z x^t + R_z x^{t-1} + b_z) \quad \text{block input} \quad (5.6)$$

$$i^t = \sigma(W_i x^t + R_i x^{t-1} + p_i \odot c^{t-1} + b_i) \quad \text{input gate} \quad (5.7)$$

$$f^t = \sigma(W_f x^t + R_f x^{t-1} + p_f \odot c^{t-1} + b_f) \quad \text{forget gate} \quad (5.8)$$

$$c^t = i^t \odot z^t + f^t \odot c^{t-1} \quad \text{cell gate} \quad (5.9)$$

$$o^t = \sigma(W_o x^t + R_o y^{t-1} + p_o \odot c^t + b_o) \quad \text{output gate} \quad (5.10)$$

$$y^t = o^t \odot h(c^t) \quad \text{block output} \quad (5.11)$$

The vector equations for a vanilla LSTM layer forward pass are given by 5.6 to 5.11.

LSTM units are widely used for language modeling and other NLP tasks. Since it has internal memory to keep information from previous steps in the sequence, it is a suitable example for a compositional function to be used in finding sentence level embeddings Young and Hazarika et el. [25]. A sentence can be modeled as a sequence of tokes or morphemes such that LSTM may capture both bag of words approach and nonlinear interactions between sentence constituents. LSTM model is superior to BI-Tanh model in taking into account not only bi-grams but all the constituents in the sentence and their interactions between them. Since LSTM networks have vanishing gradient problem during training with back propagation, the residual connections can be used to overcome this issue Boxuan et el. [26]. In LSTM models used in this study, the residual connections are used between input and cell output layer.

The third CVM model is a variant of LSTM model. It is a single layer of LSTM called ***Lstm-ScAvg***. This model has residual connections between inputs and cell outputs of each step. At the top of the model, all cell outputs are summed up and the average of cell outputs are computed.

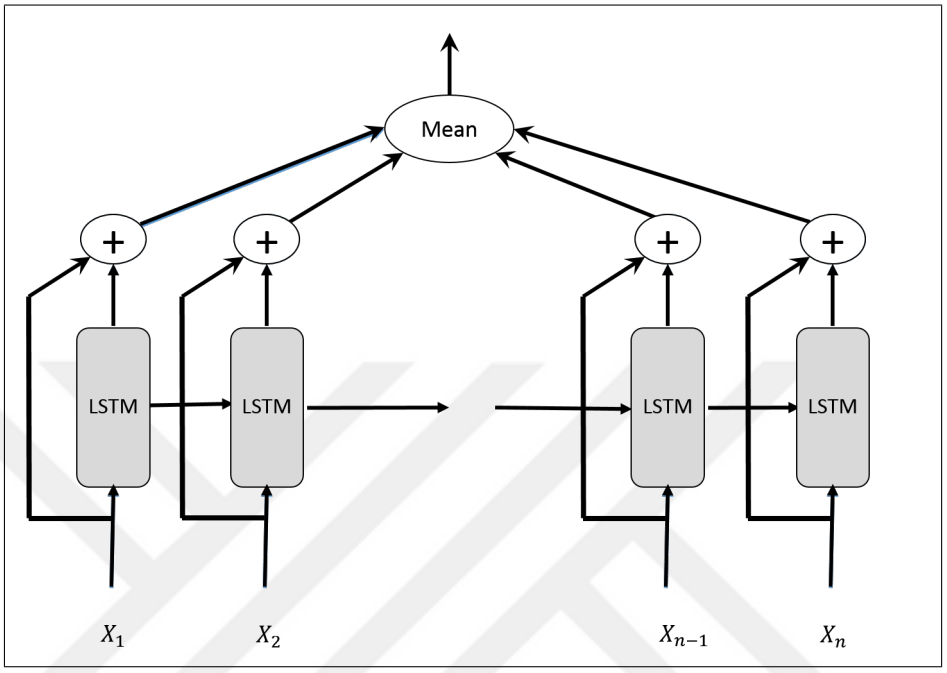

FIGURE 5.2: Layout of Lstm-ScAVG

Figure 5.2 shows the Lstm-ScAvg diagram used as a CVM in bilingual sentence representation learning. LSTM has a residual connection from input to cell outputs. At the final stage, all cell outputs are summed up and an average of them is taken.

5.2.4 BiLSTM

The fourth CVM model is a single layer bi-directional LSTM called ***BiLstm-ScAvg***. The basic idea of BiLSTM is to present each training sequence forwards and backwards to two separate LSTMs and then merge the results. This kind of network structure provides the compression and the output layers with complete past and future context for every point in the input sequence Zennaki et el. [27]. In our case, this model has residual connections between inputs and cell outputs of each step. At the top of the model, the outputs of the same sequence step from forward pass and backward pass are concatenated and summed up. The final output of the network is computed as an average of outputs of all steps.

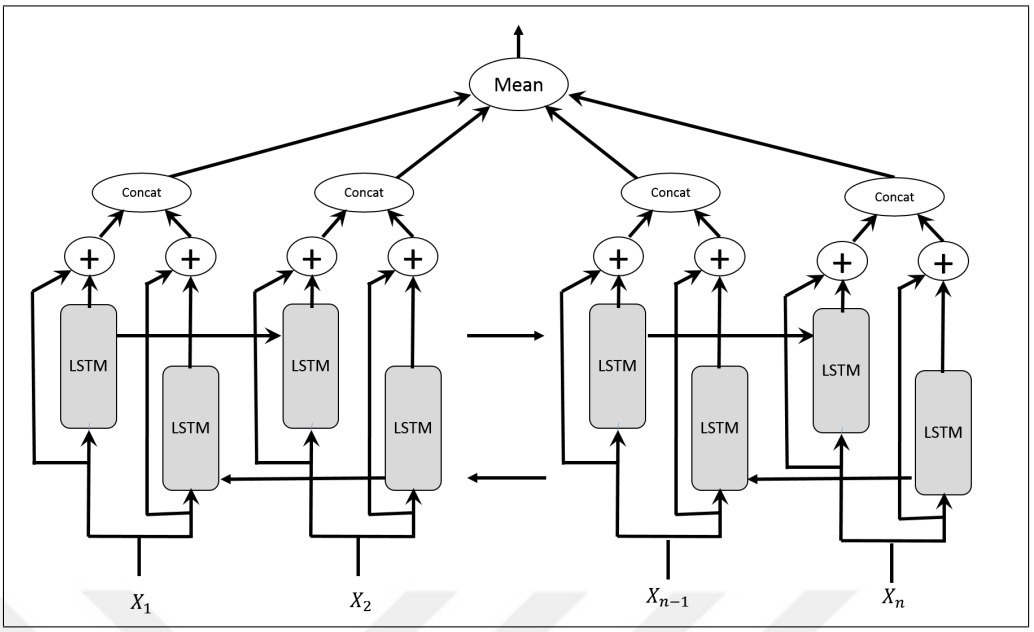

FIGURE 5.3: Layout of BiLstm-ScAVG

Figure 5.3 shows the BiLstm-ScAvg (Bidirectional LSTM) diagram used as a CVM in bilingual sentence representation learning. Bi-LSTM has residual connections from inputs to cell outputs. At the final stage cell outputs of the same steps from both direction are concatenated. Then resulting vectors from concatenation are summed up and an average of them is taken.

Chapter 6

Experiments And Tests

6.1 Corpora

We used TED corpus for IWSLT 2013 Cettolo et el. [28] [1]. This corpus contains English transcripts of all talks and their translation of different languages. All four language sets are sentence-aligned and each contains about 137000 sentences. For document classification, we used a subset of TED talks and their translations in all four languages. These talks are tagged with 15 topics to be used for document classification. For each language, topic and corpus type (token or morpheme), there are train and test sets. Morphessor by Creutz and Lagus models were trained by using Wikipedia data sets for each language [5]. Both corpora were processed using NLTK tokenizer for tokenization and Morphessor for morphological analysis. Table 1 shows the number of tokens and morpheme segments in the corpus of each language texts.

6.2 Models Setup

The following parameter list is the list of hyper-parameters of models used as CVM for learning representations.

Add model hyper-parameters are:
word vector dimension: 64
learning rate: 0.1 for the token set

[1] https://wit3.fbk.eu/

learning rate: 0.01 for the morpheme set

momentum=0.1

optimization: SGD minibatch

Bi-Tanh model hyper-parameters are :

word vector dimension: 64

learning rate: 0.01 for the token set

learning rate: 0.01 for the morpheme set

winsize=2

momentum=0.1

optimization: SGD minibatch

Lstm-ScAvg model hyper-parameters are : word vector dimension: 64

LSTM cell : 128

learning rate: 0.01,0.001,0.0001 for the token set

learning rate: 0.01,0.001,0.0001 for the morpheme set

step : 60

alpha = 0.95

dropoutProb=0.5

momentum=0.5

optimization: Rmsprop minibatch

BiLstm-ScAvg model hyper-parameters are :

word vector dimension: 64

LSTM cell : 128

learning rate: 0.01,0.001,0.0001 for the token set

learning rate: 0.01,0.001,0.0001 for the morpheme set

step : 60

alpha = 0.95

dropoutProb=0.5

momentum=0.5

optimization: Rmsprop minibatch

During training mini batch gradient descent optimization was used. The average batch size was 100. While Stochastic gradient descent was used for training Add and Bi-Tanh models, for Lstm-ScAvg and Bi-Lstm-ScAvg models, RMSPROP was used as the optimization method. Different learning rates were assigned depending on which type of the corpus was used (token or morpheme).

6.3 Hardware and Software Used For Tests

All module developments and tests were performed on TITANX 16GB memory and 12GB GPU memory with Linux OS. I have developed the modules by using Lua and Python[2]. The models were developed by using Lua and Torch deep learning library [3]. Torch has GPU support with CUDA library[4]. Because of GPU support, the deep learning models can be trained more faster than using CPU. Without GPU support, it would be almost impossible to train the models with the huge data set.

6.4 Representation learning

The following procedure was followed during the representation learning for cross-language embedding for English-German and English-French. First, a bi-lingual model for English-Turkish was trained by English-Turkish sentence-aligned parallel corpora. After constructing joint-space of embeddings for English and Turkish tokens or morphemes, at the second phase, a new bi-lingual model for English-German was trained by English-German sentence-aligned parallel corpora. The weights in the models for English set was kept fixed and the model was trained to learn embeddings for German set. At the last phase, the same procedure was applied to the English-French set. All bi-lingual models used the same CVM for each training period and the same segmentation type for the corpus. In the end, using 4 languages, 4 CVMs and 2 corpus types, 32 different models were trained and prepared for testing.

[2]all programs and data files can be found at this github address: https://github.com/sdalaman
[3]https://github.com/torch/nn
[4]https://github.com/torch/cutorch

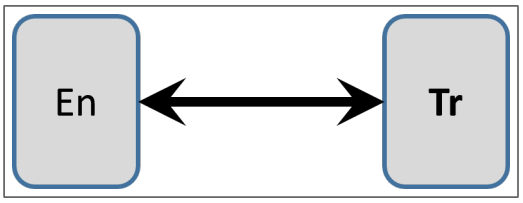

FIGURE 6.1: Lstm bi-lingual model for English-Turkish parallel corpora. The weights of both models are updated.

Figure 6.1 shows learning process of a bi-lingual model for English-Turkish parallel corpora. The models updated the weights of both language sides during training by the parallel corpus English-Turkish. The bi-lingual model was trained to find the proper weights of both sides such that the corresponding sentence vectors from English and Turkish sets were aligned adjacent to each other.

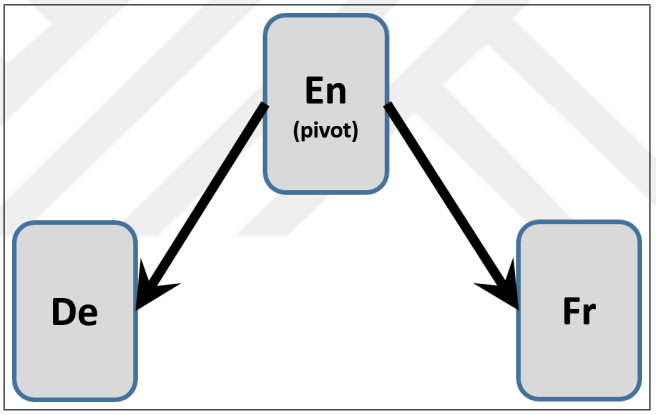

FIGURE 6.2: Multilingual model training with English-German and English-French corpora

Figure 6.2 shows learning process of two bi-lingual models trained with English-German and English-French corpora respectively. Model weights for English set were used as pivot reference and kept fixed. On the other hand, the model weights for German and French sets were updated to align the corresponding vectors as much as adjacent to each other and the English counterpart.

Table 6.1: Statistics for Corpus Segmentation

Language

Segmentation	English	German	French	Turkish
Number of Tokens	55885	106855	79951	184475
Number of Morphemes	23664	37812	28728	48721

As indicated before, two type of segmentation models were tested. Table 6.1 shows the number of tokens and morphemes for each language. In each language corpus, there are approximately 138000 sentences as can be seen in Table 6.2

Table 6.2: Number of sentences for each language-pair corpus

Language

Number of sentences	German	French	Turkish
English	138500	154500	136796

6.5 Paraphrase Tests

Two experiments were performed and reported during paraphrase test. First, we evaluated models with paraphrase tests. Randomly selected 100 sentences from English set and their corresponding pair sentences from other languages set were compared according to the closeness of their embedding vectors. If the corresponding sentence was the closest one, the score was increased by one. The total score of these comparisons was reported. This test evaluates how well models compute sentence embeddings for the corresponding sentence pairs from each language. This test was performed for token and morpheme segmented corpora with all 4 CVM models. Table 6.3 and 6.4 summarize their results.

Table 6.3: Paraphrase Test Results of Corpus Split into Tokens

Language Direction	CVM	German	French	Turkish
En->L2	Add	**0.4348925411**	**0.8310589162**	**0.6417025063**
	BI-Tanh	0.0611883692	0.0709224511	0.5043749073
	Lstm-ScAvg	0.2371638142	0.3545543585	0.2190150801
	BiLstm-ScAvg	0.2621230644	0.3919360104	0.2173656927
L2->En	Add	**0.1730720607**	**0.6803569085**	**0.6378466558**
	BI-Tanh	0.0987357775	0.258037003	0.5343318997
	Lstm-ScAvg	0.157192339	0.2965502231	0.218072573
	BiLstm-ScAvg	0.1365118174	0.3147241267	0.1849670123

Table 6.4: Paraphrase Test Results of Corpus Split into Morphemes

Language Direction	CVM	German	French	Turkish
En->L2	Add	**0.3032869785**	**0.522700433**	**0.4395669583**
	BI-Tanh	0.0499367889	0.0856186852	0.3692718375
	Lstm-ScAvg	0.2442950285	0.4388943302	0.271795476
	BiLstm-ScAvg	0.1145069275	0.1856023506	0.1240574929
L2->En	Add	0.1403286979	**0.3763285658**	**0.3977458105**
	BI-Tanh	0.0798988622	0.0828631413	0.3916654308
	Lstm-ScAvg	**0.1485330073**	0.3203830667	0.2721489161
	BiLstm-ScAvg	0.0634678077	0.1628577647	0.1237040528

6.5.1 Paraphrase Test Results Discussion

Test results show percentage of success in finding correct L1-L2 sentence pairs. Tests were performed in both directions. The first is English sentence against the other language L2 and the second is the other languages sentence against English sentence. Bold indicates best results. According to the test results, the most successful CVM is *Add* method for both token and morpheme sets . For token set, *Lstm-ScAvg* and *BiLstm-ScAvg* are close to each other and they are far better than *BiTanh* except Turkish set. However, for morpheme set, *Lstm-ScAvg* is better than *BiLstm-ScAvg*. The worst performance belongs to *BiTanh* for German and French set. This is probably due to their sensitivity to the order of words in a sentence. The performance of *BiTanh* on Turkish set is far better than other language sets because the meaning of a sentence is less sensitive to the order of words compared to others in Turkish language.

6.6 Cross-Lingual Document Classification (CLDC) Tests

There are documents of 4 languages with 15 different topic tags. For each topic, we have two training sets. One of them is tagged as the positive set, which contains the documents that belong to the topic. The other one is tagged as the negative set, which contains the documents that do not belong to the topic. For each topic, a classifier was trained to differentiate between negative and positive examples. Training of a classifier for a topic was done with a set of training documents from language L1 and tested with a set of test documents of Language L2. For this purpose, for each language, a training and test set of 15 topics were prepared.

The primary target in CLDC test is to use joint-space embeddings for classification. The representations of language constituents (words, tokens, and morphemes) were learned by using a parallel corpus. Consequently embedding vectors of corresponding hierarchical structures like a sentence or document for all languages would be adjacent to each other in the joint-space. Then the document classifier can use this adjacency such that it can be trained with data from high-resource language and used to classify the documents from a low-resource language.

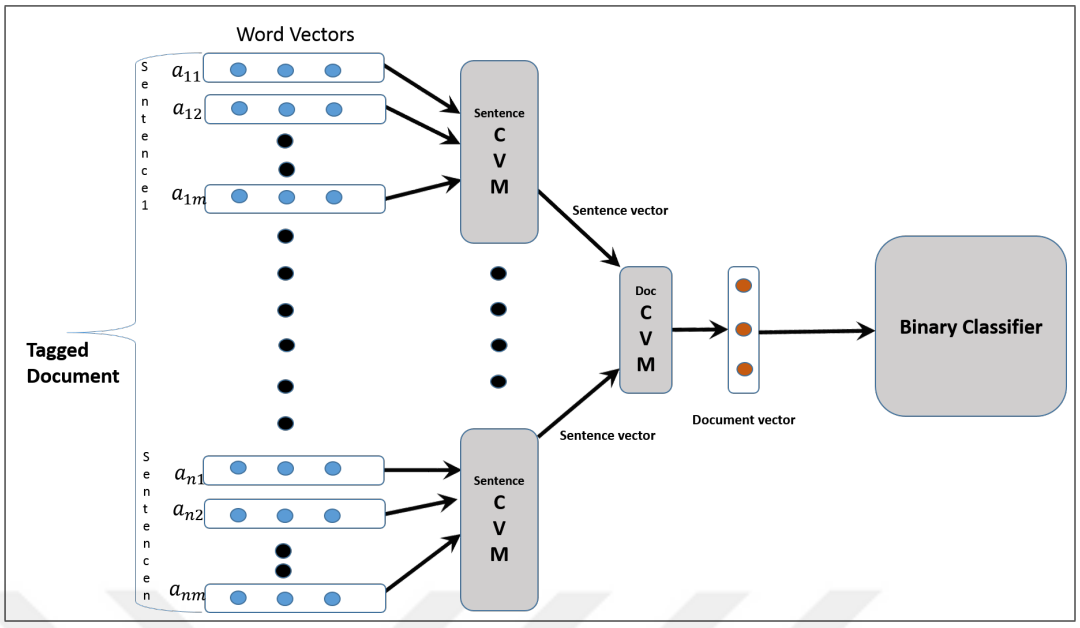

FIGURE 6.3: Document Classifier Diagram

Figure 6.3 shows the combined diagram for CVM and a binary classifier. The classifier uses a CVM model to compute sentence embedding vectors. Then document embeddings are computed by summing all embedding vectors of sentences in the document. The document vector is used as input to the binary classifier.

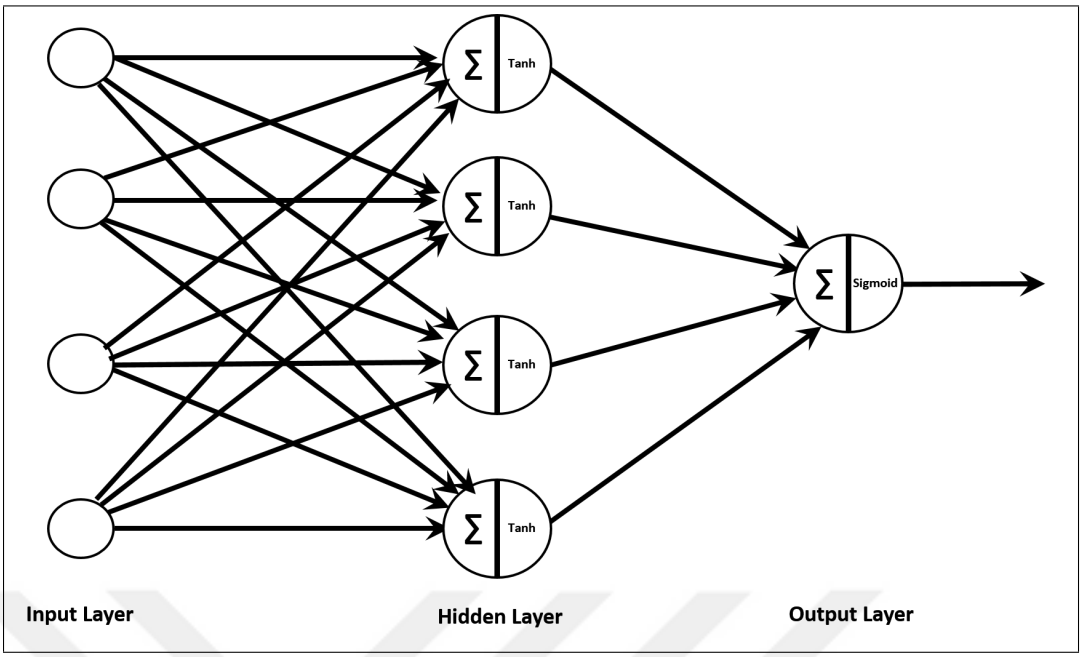

FIGURE 6.4: Binary Classifier Diagram

Figure 6.4 shows binary document classifier model with one hidden layer, which is used for CLDC test. Hidden layer is using ***Tanh*** activation function to squash the aggregation of embedding values of document vector. ***Sigmoid*** function is used in the output layer to differentiate between positive and negative samples.

The classifiers are built upon the sentence embedding models. The training and evaluation of a classifier are as follows: first, a document is prepared either with tokenization or morphological analysis according to the model type. Then by using a token or morpheme embeddings and CVM model, an embedding vector is computed for each sentence in the document. Then all sentence embeddings are added to find an embedding vector for a document. After computing document embedding vectors of all the documents in the training set, the classifier is trained using document embedding vectors of positive and negative samples. During this step, a CVM model is employed to compute sentence embeddings before classifier training. By using these steps, we trained each classifier with the train set and tested with the test set of the corresponding topic. Train and test samples were strictly separate sets. After executing the training and testing procedure 10 times, we reported the average accuracy and F1-score for each topic. Finally, as a test result, we reported the average accuracy score and F1-score of all topic classifiers. Training sample documents were selected from the set of high-resource language and test sample documents were from other languages sets.

Table 6.5: Classifier F1-Scores of Corpus Split into Tokens

Language Direction	CVM	German	French	Turkish
En->L2	Add	**0.2078455416**	**0.381443812**	**0.328651887**
	BI-Tanh	0.106378589	0.158658606	0.231282439
	Lstm-ScAvg	0.190603733	0.343417603	0.321859386
	BiLstm-ScAvg	0.1991546494	0.1292028734	0.1516834759
L2->En	Add	**0.2530711924**	**0.3238294718**	0.299052239
	BI-Tanh	0.087920122	0.16652522	0.211125728
	Lstm-ScAvg	0.25608471	0.28965102	**0.308054251** -
	BiLstm-ScAvg	0.1344384198	0.1939823552	0.2143356161

Table 6.6: Classifier Acc-Scores of Corpus Split into Tokens

Language Direction	CVM	German	French	Turkish
En->L2	Add	50.6153846154	72.5850340136	76.3636363636
	BI-Tanh	28.81538462	47.42857143	67.68484848
	Lstm-ScAvg	**68.34871795**	**72.80272109**	**77.38181818**
	BiLstm-ScAvg	55.3230769231	29.7142857143	34.3272727273
L2->En	Add	60.4242424242	**75.5151515152**	56.1212121212
	BI-Tanh	25.88484848	48.57575758	65.81818182
	Lstm-ScAvg	**65.81818182**	68.29090909	**71.34545455**
	BiLstm-ScAvg	33.6242424242	55.7212121212	54.7757575758

Table 6.7: Classifier F1-Scores of Corpus Split into Morphemes

Language Direction	CVM	German	French	Turkish
En->L2	Add	0.1329648119	0.370417242	0.3181007909
	BI-Tanh	**0.189251966**	0.121199969	0.119846185
	Lstm-ScAvg	0.182662077	**0.3758209**	**0.327705446**
	BiLstm-ScAvg	0.1876407684	0.22134477796	0.1863615868
L2->En	Add	0.1837960883	0.21629	0.2424135911
	BI-Tanh	0.109304131	0.10261755	0.192656384
	Lstm-ScAvg	**0.303459923**	**0.327898993**	**0.345371767**
	BiLstm-ScAvg	0.1925581865	0.1909009205	0.2071623678

TABLE 6.8: Classifier Acc-Scores of Corpus Split into Morphemes

Language Direction	CVM	German	French	Turkish
En->L2	Add	**70.6153846154**	79.1156462585	**76.4848484848**
	BI-Tanh	17.3538461538	20.306122449	40.0727272727
	Lstm-ScAvg	46.63589744	**79.63265306**	75.92727273
	BiLstm-ScAvg	52.3076923077	54.3809523810	48.9212121212
L2->En	Add	42.4848484848	56.3636363636	66.7272727273
	BI-Tanh	35.05454545	27.1030303	42.18787879
	Lstm-ScAvg	**67.9030303**	**74.41212121**	**78.32727273**
	BiLstm-ScAvg	49.1636363636	49.4181818182 6	51.2363636364

Table 6.9: Cross-Lingual Classifier Acc Scores of Corpora Split into Tokens

CVM=Add (Token)	Test Language		
Training Language	German	French	Turkish
German		65.93197279	68.01212121
French	55.37435897		**70.93333333**
Turkish	61.85641026	69.76870748	

Table 6.10: Cross-Lingual Classifier Acc Scores of Corpora Split into Morphemes

CVM=Add (Morpheme)	Test Language		
Training Language	German	French	Turkish
German		57.89115646	62.24242424
French	60.87179487		72.36363636
Turkish	63.12820513	**73.80952381**	

Table 6.11: Cross-Lingual Classifier Acc Scores of Corpora Split into Tokens

CVM=Bi-Tanh (Token)	Test Language		
Training Language	German	French	Turkish
German		35.31972789	30.2969697
French	20.56923077		**65.3636363**
Turkish	13.8615384	50.53061224	

Table 6.12: Cross-Lingual Classifier Acc Scores of Corpora Split into Morphemes

CVM=Bi-Tanh (Morpheme)	Test Language		
Training Language	German	French	Turkish
German		13.11564626	19.6
French	20.77948718		**29.65454545**
Turkish	17.13333333	20.53741497	

Table 6.13: Cross-Lingual Classifier Acc Scores of Corpora Split into Tokens

CVM=Lstm-ScAvg (Token)	Test Language		
Training Language	German	French	Turkish
German		65.93197279	68.01212121
French	55.37435897		**70.93333333**
Turkish	61.85641026	69.76870748	

Table 6.14: Cross-Lingual Classifier Acc Scores of Corpora Split into Morphemes

CVM=Lstm-ScAvg (Morpheme)	Test Language		
Training Language	German	French	Turkish
German		69.79591837	67.39393939
French	67.90769231		69.13939394
Turkish	58.87179487	**75.55102041**	

Table 6.15: Cross-Lingual Classifier Acc Scores of Corpora Split into Tokens

CVM=BiLstm-ScAvg (Token)	Test Language		
Training Language	German	French	Turkish
German		35.4013605442	35.5636363636 -
French	33.1076923077		38.496969697
Turkish	**53.9076923077**	53.6054421769	

Table 6.16: Cross-Lingual Classifier Acc Scores of Corpora Split into Morphemes

CVM=BiLstm-ScAvg (Morpheme)	Test Language		
Training Language	German	French	Turkish
German		51.48979592	49.9030303
French	49.33333333		**51.75757576**
Turkish	48.96410256	48.2585034	

6.6.1 CLDC Test Results Discussion

The test results were reported for F1-score and accuracy rate for each CVM model and corpus segmentation type. CLDC tests were performed in both direction (training with English documents, tested with L2 documents or vice versa). Bold indicates best results. In CLDC, English-L2 or L2-English tests, *Lstm-ScAvg* CVM F1 scores are better than others. However, morpheme set results of English-L2 or L2-English tests for *Add* and *Lstm-ScAvg* are close to each other compared to the other CVM results. Cross-lingual test result between languages except English set shows that Turkish and French representation are close to each other compared to others. *Add* and *Lstm-ScAvg* performances are better than others.

The overall performance of *Add* CVM is better than others. *Lstm-ScAvg* is very close to *Add*. The training time is very long for *Lstm-ScAvg* and *BiLstm-ScAVG* because of their network structure. Although *Add* model seems to have better performance, the main

reason for this is that the order of words in a sentence is not the critical characteristic of these languages. *Add* model may not be successful with some other languages where the order is determinate. Lstm models are proven to perform well at modeling ordered sequences. Lstm model and its derivatives can be more successful to model semantic and syntactic representation of a sentence because sentence constituents can be considered as a sequence and their order is both important and decisive. The main drawbacks of Lstm models can be listed as their memory requirement, which is much higher than other simple models and they need huge data set to reach more satisfying performance increase.

Chapter 7

Summary And Conclusion

In this study, we have presented the applications and performance of compositional vector models combined with token and morpheme segmentation methods for learning multilingual representations. The distributed representation of tokens and morphemes were computed by using sentence-aligned parallel corpora. Parallel corpora for four languages English, Turkish, German and French were used. All embeddings in the joint-space were computed in conjunction with a multilingual objective function for compositional vector models.

With this approach, we investigated how to find representations of different language constituents such as tokens and morphemes in multilingual joint-space and their relative effect on the performance of an NLP task like document classification. Moreover, we examined the embeddings learned with different composition functions and their impact on the performance of cross-lingual document classification.

It has been shown that multilingual language embeddings can be a useful tool for different NLP tasks where multilingual performance is expected. For example, this method can be an effective way to improve the performance of document classification models for a language with low resources by transferring the information from a language with high resources.

Bibliography

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=944919.944966`.

[2] N. Kalchbrenner and P. Blunsom. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the ACL 1st. Workshop on Continuous Vector Space Models and their compositionality*. ACL, 2013.

[3] K. M. Hermann and P. Blunsom. The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

[4] M. Creutz and K. Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):3:1–3:34, February 2007. ISSN 1550-4875. doi: 10.1145/1187415.1187418. URL `http://doi.acm.org/10.1145/1187415.1187418`.

[5] M. Creutz and K. Lagus. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning - Volume 6*, MPL '02, pages 21–30, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118647.1118650. URL `https://doi.org/10.3115/1118647.1118650`.

[6] S Ruder. A survey of cross-lingual embedding models. arxiv preprint arxiv:1706.04902v1. 2017.

[7] K. M. Hermann and P. Blunsom. Multilingual Distributed Representations without Word Alignment. In *Proceedings of ICLR*, April 2014. URL `http://arxiv.org/abs/1312.6173`.

[8] S Lauly, A Boulanger, and H Larochelle. Learning multilingual word representations using a bag-of-words autoencoder. *CoRR*, abs/1401.1803, 2014.

[9] K. M. Hermann T. Kocisky and P. Blunsom. Learning bilingual word representations by marginalizing alignments. 2014.

[10] A. S. Chandar, S. Lauly, H. Larochelle, M. Khapra, B. Ravindran, V. C. Raykar, and A. Saha. An autoencoder approach to learning bilingual word representations. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1853–1861. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5270-an-autoencoder-approach-to-learning-bilingual-word-representations.pdf`.

[11] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org, 2014. URL `http://dl.acm.org/citation.cfm?id=3044805.3045025`.

[12] H. Pham, T. Luong, and C. Manning. Learning distributed representations for multilingual text sequences. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 88–94. Association for Computational Linguistics, 2015. doi: 10.3115/v1/W15-1512. URL `http://www.aclweb.org/anthology/W15-1512`.

[13] G. Sohsah. Multilingual distributed word representation using deep learning. Master's thesis, İstanbul Şehir University, Istanbul, Turkey, 2016.

[14] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390177. URL `http://doi.acm.org/10.1145/1390156.1390177`.

[15] A. Mnih and G. Hinton. A scalable hierarchical distributed language model. In *In NIPS*, 2008.

[16] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010. URL `http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html`.

[17] K. Erk and S. Padó. A structured vector space model for word meaning in context. In *EMNLP*, 2008.

[18] J .Mitchell and M .Lapata. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, 2008.

[19] S. Clark and S. Pulman. Combining symbolic and distributional models of meaning. *Proceedings of the AAAI Spring Symposium on Quantum Interaction*, pages 52–55, 2007. URL `http://www.aaai.org/Papers/Symposia/Spring/2007/SS-07-08/SS07-08-008.pdf`.

[20] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL `http://dl.acm.org/citation.cfm?id=2145432.2145450`.

[21] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1201–1211, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=2390948.2391084`.

[22] N. Kalchbrenner and P. Blunsom. Recurrent convolutional neural networks for discourse compositionality. *Proceedings of the 2013 Workshop on Continuous Vector Space Models and their Compositionality*, 2013.

[23] K. M. Hermann and P. Blunsom. Multilingual Models for Compositional Distributional Semantics. In *Proceedings of ACL*, June 2014. URL `http://arxiv.org/abs/1404.4641`.

[24] J. Koutnik B. R. Steunebrink J. Schmidhuber K. Greff, R. K. Srivastava. Lstm: A search space odyssey. 2015.

[25] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent Trends in Deep Learning Based Natural Language Processing. *ArXiv e-prints*, August 2017.

[26] B. Yue, J. Fu, and J. Liang. Residual recurrent neural networks for learning sequential representations. *Information*, 9(3), 2018.

[27] L. Besacier O. Zennaki, N. Semmar. Inducing multilingual text analysis tools using bidirectional recurrent neural networks.arxiv preprint arxiv:1609.09382v1. 2014.

[28] M. Cettolo, C. Girardi, and M. Federico. Wit3: Web inventory of transcribed and translated talks. pages 261–268, 01 2012.