

Multilingual Distributed Word Representation Using Deeplearning

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

by

Gihad SOHSAH

in partial fulfillment for the
degree of Master of Science

in

Electronics and Computer Engineering



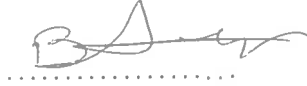
This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Electrical and Computer Engineering.

APPROVED BY:

Assist. Prof. Dr. Onur Güzey
(Thesis Advisor)



Assist. Prof. Dr. Barış Arslan



Assist. Prof. Dr. Arzucan Özgür



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: 26 September 2016

SEAL/SIGNATURE:



Declaration of Authorship

I, Gihad SOHSAH, declare that this thesis titled, 'Multilingual Distributed Word Representation Using Deeplearning' 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:

Gihad Sohsah

Date:

26/9/2016

“He who does not know how to look back at where he came from will never get to his destination.”

José Rizal



Multilingual Distributed Word Representation Using Deeplearning

Gihad SOHSAH

Abstract

In this work, the problem of extracting meaningful multilingual word embeddings is studied with special focus on morphologically-rich languages. In order to achieve multilingualism, a data-driven method that makes use of a sentence-aligned parallel corpus is used. This method is expanded hierarchically to take account for the words parts, tokens or morphemes, rather than just considering the raw words as the basic language units. Also various architectures and aggregation functions for constructing word and sentences embeddings given their parts are studied and compared. The aggregation function for a specific function is chosen according to the nature of the particular language. To evaluate the different methods, one sanity check test is used and two more tests are used to evaluate the quality of the resulting representations. The sanity check which is mainly used to make sure that the models are learning anything from the corpus and is also used for the parameter tuning, is the paraphrase test. The second test, t-SNE, is a visual test that just gives insights about the model's ability to bring semantically equivalent words and sentences close to each other in the space. The third test, that gives the most meaningful measure, is the cross-lingual document classification task. The (CLDC) task is concerned with training a supervised classifier using documents from one language, the rich-resources language, and testing it using another language, the low-resources one, while maintaining a satisfying performance. The performances of the models are described in terms of the F1-score. As the experiments have shown, a multilingual framework for extracting word-embeddings jointly for both English and Turkish that uses additive functions at both sentences and words level results the best result in terms of the F1-score achieved in the (CLDC) task. Various data preprocessing methods are also studied, the words can be either extracted by simply dividing the sentences using the space as a delimiter, using a tokenizer, or using a morphological analyzer in the case of the morphologically-rich languages. The experiments showed that using the raw data format for English and the morphemes as the basic language unit for Turkish yields the best results.

Keywords: Natural Language Processing, Deeplearning, Neural Networks, Morphologically-rich languages, Turkish NLP Natural Language Processing, Language Model, Deeplearning, Neural Networks, Multilingual, Word Embeddings

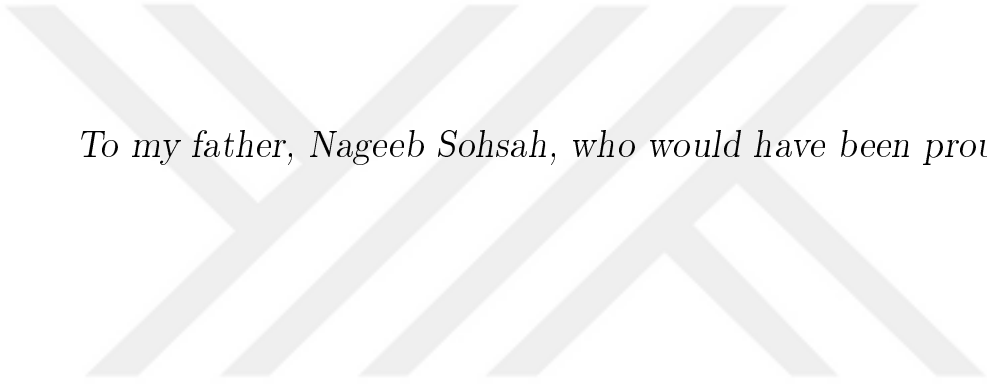
Derin öğrenme ile çok dilli, dağıtılmış kelime temsilleri

Gihad SOHSAH

ÖZ

Bu çalışmada manalı çok dilli temsillerin çıkarılması problemi morfolojik olarak zengin dillere odakla incelenmiştir. Çok dilliliği sağlamak için cümle olarak eşleştirilmiş metinleri kullanan veri tabanlı bir metot kullanılmıştır. Bu metot sadece yalın kelimeleri değil, hiyerarşik olarak kelime parçaları, ekleri ve diğer morphemeleri dikkate alacak şekilde hiyerarşik olarak geliştirilmiştir. Ayrıca, parçalarından kelime ve cümle temsilleri oluşturmak için farklı mimari ve birleştirme fonksiyonları incelenmiş ve karşılaştırılmıştır. Bir dilin birleştirme fonksiyonu o dilin özelliklerine göre seçilmiştir. Farklı metotları karşılaştırmak ve temsillerin amaca uygunluğunu belirlemek için, biri hızlı kontrol amaçlı olmak üzere üç test kullanılmıştır. Hızlı kontrol için oluşturulan paraphrase testi hem metinlerden öğrenme olup olmadığını kontrol etmek, hem de parametreleri belirlemek için kullanılmıştır. İkinci test, t-SNE, modelin birbirine eşit kelime ve cümleleri bir araya getirmesinin görsel bir testidir. En anlamlı sonucu veren üçüncü test bir diller arası döküman sınıflandırma (CLDC) testidir. Bu test çok kaynak bulunan bir dilde eğitilen gözetimli öğrenme sınıflandırıcının az kaynak bulunan bir dilde performans kaybı olmadan test edilmesiyle ilgilidir. Modellerin performansı recall ve precision'ı göz önünde bulunduran bir ölçü olan F1-skoru ile açıklanmaktadır. Bu çalışmada kelime temsilleri kullanılarak İngilizce gibi çok kaynaklı bir dilden, Türkçe gibi az kaynaklı bir dile bilgi aktarılması ve İngilizce üzerinde eğitilen modellerin Türkçe dökümanların sınıflandırılmasında yeterli performans sağlamasına odaklanılmıştır. Bu test de, averaged-perceptron sınıflandırıcısı TED-Corpus kullanılarak eğitilmektedir. Bu sınıflandırıcı eğitildikten sonra bir dökümanı temsiline göre 14 sınıftan birine atamalıdır. Bu döküman temsili bütün kelimelerin temsillerini ekleyerek oluşturulmaktadır. Averaged perceptron iki sınıflı bir sınıflandırıcı olduğundan one-vs-all tekniği kullanılarak bu sınıflandırıcı 14 sınıflı bir sınıflandırıcıya dönüştürülmüştür. Deneyle göre, kelime temsillerini Türkçe ve İngilizce için ortak çıkaran ve toplama kompozisyon fonksiyonlarını kullanan çok dilli yöntem, hem kelime hem cümle seviyesinde CLDC F1 skoru olarak en iyi sonuçları vermiştir. Farklı veri işleme yöntemleri incelenmiştir. Cümleler bir tokenizer kullanarak boşluklar ile kelimelere yada zengin morfolojisi olan diller için bir morfolojik inceleyici kullanılarak morphemelerine ayrılabilir. Deneysel sonuçlarına göre İngilizce'de kelimeleri, Türkçe'de morphemeleri kullanmak en iyi sonuçları vermiştir.

Anahtar Sözcükler: Doğal dil işleme, derin öğrenme, sinir ağları, Zengin morfolojisi olan diller, Türkçe doğal dil işleme Mühendislik, Deneysel Psikoloji



To my father, Nageeb Sohsah, who would have been proud . . .

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor Prof. Onur Güzey for the continuous support during my journey as a student at Şehir University. All thanks are due to him ...for his patience, motivation, and knowledge. Without his guidance I would not have made it to this point. He has been a great mentor as the great mentor should be!

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 incited me to widen my research from various perspectives.

I thank my fellow labmates in for the stimulating discussions, for being a great company during the sleepless nights in the lab. Also I thank my friends for being there whenever I needed support. Last but certainly not least, I would like to thank my family: my parents, my brother and little sister for supporting me spiritually throughout my entire life.

Contents

Abstract	iv
Öz	v
Acknowledgments	vii
List of Figures	x
List of Tables	xii
1 Background	1
1.1 Introduction	1
1.2 Dimensionality Reduction Methods	1
1.2.1 Word co-occurrence matrix	1
1.2.1.1 Principal Component Analysis	2
1.2.1.2 Pointwise Mutual Information Matrix	3
1.2.1.3 Skip-Gram with Negative Sampling	3
1.3 Neural Networks Methods	4
1.3.1 SENNA embeddings	4
1.3.2 Turian embeddings	5
1.3.3 HLBL embeddings	6
1.3.4 Huang’s embeddings	6
1.3.5 Word2vec	7
1.3.6 Polyglot	7
1.4 Cross Lingual Models	8
1.4.1 Klementiev et al. embeddings	8
1.4.1.1 Multitask Learning	9
1.4.1.2 The Neural Network Model	9
1.4.2 Karl Moritz Hermann and Phil Blunsom embeddings	10
1.5 Problem Definition	12
1.5.1 Available methods for tackling the proposed issues	12
1.5.1.1 Using Simple Additive Function	13
1.5.1.2 Using Recurrent Neural Network	13
2 Proposed Method	15
2.1 Introduction	15
2.2 Proposed Framework	15

3	Experiments and Results	19
3.1	Introduction	19
3.1.1	Experimental Setting	19
3.1.1.1	Choosing the composition function	19
3.1.1.2	Data preprocessing	20
3.1.1.3	Training and Parameter Tuning	21
3.1.1.4	The Paraphrase Test	22
3.1.1.5	The CLDC Test	22
3.1.1.6	Hardware and Software	23
3.1.2	Experimental Results	23
3.1.2.1	Extracting the Embeddings	24
3.1.2.2	Evaluating the quality of the embeddings	24
4	Conclusion	37
4.1	Introduction	37
4.2	Conclusion	37
	Bibliography	40

List of Figures

1.1	The neural network architecture proposed by Collobert and Weston [8] for extracting word embeddings using a fixed-sized window. The same architecture was also exploited by Turian et al. [23] with few differences in the experimental settings.	5
1.2	The architecture proposed by Huang et al. [12] for extracting word embeddings using both local and global contexts to capture the multiple meanings of the same word by learning from the global context i.e. document words. The model uses the same architecture proposed by Turian et al. [23].	7
1.3	The architecture proposed by Bengio et al. [3] for extracting word embeddings using $n - gram$ sequences. A similar architecture is used by Klementiev et al. [13] to train multiple models for multiple languages jointly using MTL described in section	10
1.4	The bilingual architecture proposed by Hermann and Blunsom [11] for extracting word embeddings leveraging parallel corpora.	11
1.5	The layered architecture proposed by Smith and Eisner [22] that utilizes RNN to build word embeddings from morphemes embeddings.	14
2.1	English-Turkish framework for extracting word embeddings	17
3.1	t-SNE visualization of animals names in Turkish and English using word embeddings extracted by the 6-gram tanh model using tokenization as a preprocessing method	25
3.2	t-SNE visualization of numbers in Turkish and English using word embeddings extracted by the 6-gram tanh model using tokenization as a preprocessing method	26
3.3	t-SNE visualization of sentences in Turkish and English using word embeddings extracted by the 6-gram tanh model using tokenization as a preprocessing method	27
3.4	t-SNE visualization of animals names in Turkish and English using word embeddings extracted by the 6-gram tanh model using raw words	28
3.5	t-SNE visualization of numbers in Turkish and English using word embeddings extracted by the 6-gram tanh model using raw words	29
3.6	t-SNE visualization of sentences in Turkish and English using word embeddings extracted by the 6-gram tanh model using raw words	30
3.7	t-SNE visualization of animals names in Turkish and English using word embeddings extracted by the additive model using raw words	31
3.8	t-SNE visualization of numbers in Turkish and English using word embeddings extracted by the additive model using raw words	32

3.9	t-SNE visualization of sentences in Turkish and English using word embeddings extracted by additive model using raw words	33
3.10	t-SNE visualization of animals names in Turkish and English using word embeddings extracted by the additive model using tokenization as a preprocessing method	34
3.11	t-SNE visualization of numbers in Turkish and English using word embeddings extracted by the additive model using tokenization as a preprocessing method	35
3.12	t-SNE visualization of sentences in Turkish and English using word embeddings extracted by the additive model using tokenization as a preprocessing method	36



List of Tables

3.1	The paraphrase test results for each type of models studied in this work according to the data preprocessing scheme using add function as a composition function.	24
3.2	The paraphrase test results for each type of models studied in this work according to the data preprocessing scheme using n-gram tanh function as a composition function.	24
3.3	F1-scores obtained by training four models using four types of embeddings. The language of the training set is English while the language for the test set is Turkish.	25

Chapter 1

Background

1.1 Introduction

Distributed word representation, also known as word embedding, is a set language modelling and feature representation techniques for various natural language processing tasks. They also can be defined as a set of low-dimensional vector space where each vector represents a word and the dimensions are the potential features to describe the semantic and syntactic properties of the word. It is a low-dimensional space compared to the number of the words in the vocabulary. Each word is described in the space as a continuous real-valued vector such that with similar semantics, syntactic properties, or share the same context in the corpus should be in proximity to each other in the space. Word-embedding can be extracted from huge corpora using various methods, including neural networks, dimensionality reduction methods, and even probabilistic methods. In the following subsections we overview the various methods for extracting word embedding.

1.2 Dimensionality Reduction Methods

In this set of methods, word embeddings are extracted by running dimensionality reduction algorithms on the word-context or words co-occurrence matrix of the corpus. These methods are easier and less time-consuming than the neural methods which need long time and tons of data for training.

1.2.1 Word co-occurrence matrix

The word-context matrix, or word co-occurrence matrix, is formed by counting the occurrences of the words in the contexts. For example, for the corpus C we have two

main sets, 1) V_w which represents the words vocabulary. 2) V_c which represents the contexts vocabulary. Each c belongs to V_c is an L -sized window, and the context of any word is all the surrounding words. The co-occurrence matrix is formed by computing all probabilities $P(w | c)$.

$$P(w | c) = \frac{P(w, c)}{P(c)} = \frac{n(w, c)}{\sum_w n(w, c)} \quad (1.1)$$

The co-occurrence matrix size is dependent on the number of words in the vocabulary, so using the discrete distribution directly as word embeddings is not practical for large vocabulary. That is why after forming the word co-occurrence matrix, a dimensionality reduction algorithm, such as Principal Component Analysis, can be used to extract the word embeddings.

A method that exploits Hellinger PCA as a technique for extracting word embeddings from words co-occurrence matrix is described in Lebet and Collobert [14]. In their work, they use Hellinger distance to compute word discrete distributions. Afterwards they use PCA to reduce the dimensions in order to get the final word embeddings. They show that a simple spectral method as PCA can generate word embeddings that are at least as good as the ones extracted using deep-learning architectures.

1.2.1.1 Principal Component Analysis

Principal Component Analysis, PCA, was first formulated in statistics by Pearson [21], who formulated the analysis as finding "lines and planes of closest fit to systems of points in space". As illustrated by Wold et al. [24], PCA provides a way of reducing a multivariate data matrix X as the product of two small matrices T and P^T . These matrices, T and P^T are believed to capture the essential data represented by the matrix X . By plotting the columns of T , we can get a picture of the dominant "object patterns" and by plotting the rows of P^T we can get a picture of the dominant "variable patterns". PCA can be used for various tasks, including: 1) simplification, 2) data reduction and compression, 3) modelling, and 4) feature-selection for machine learning problems. In this context, PCA is used to reduce the data and detect the correlating dimensions given the word-context or the words co-occurrence matrix and then map the data into a space of a set of linearly uncorrelated dimensions called *principal components*.

Another work by Levy and Goldberg [15] tackles the problem as an implicit matrix factorization problem. They consider the words co-occurrence matrix and try to broaden

the understanding of the problem. They argue that the training objective of the neural-network language modelling is to maximize the dot-product between the vectors of word-context pairs that frequently co-occur. On the other hand, it is desirable to minimize the dot-product for random word-context pairs. In their work, they try to model Skip-Gram with Negative Sampling known as, SGNS, training method as weighted matrix factorization problem. They show that the objective of the training is implicitly factorizing a shifted Pointwise Mutual Information matrix. In the following subsections we give further details about PMI matrix, and how to model the objective of SGNS.

1.2.1.2 Pointwise Mutual Information Matrix

From information theory, pairwise mutual information is an association measure between two discrete outcomes p and q , it can be defined as:

$$PMI(x, y) = \log \frac{P(x, y)}{P(x) \cdot P(y)} \quad (1.2)$$

In the NLP context, $PMI(w, c)$ is an association measure between a word w and a context c . the use of PMI as an association measure was firstly introduced by Church and Hanks [7]. The PMI for word-context can be estimated using the counts of occurrences as follows:

$$PMI(w, c) = \log \frac{\text{count}(w, c) \cdot |C|}{n(w) \cdot n(c)} \quad (1.3)$$

An extension to this definition to make it more fit for NLP problems is the positive PMI metric. Where all negative entries in the matrix are replaced by zero. Or more formally, for each (w, c) pair we do the following:

$$PPMI(w, c) = \max(PMI(w, c), 0) \quad (1.4)$$

1.2.1.3 Skip-Gram with Negative Sampling

In Skip-Gram with negative sampling SGNS introduced in Mikolov et al. [17] the objective can be defined as follows:

Consider a word context pair (w, c) and a corpus C . let $P(C = 1 | w, c)$ be the probability that (w, c) is a valid pair in the corpus, i.e. there is actually an L-sized window c in

the corpus C that contains the word w . And let be the probability that it is not a valid pair in the corpus.

The negative sampling objective tries to maximize $P(C = 1 | w, c)$ for all (w, c) pairs in the corpus while also maximizing $P(C = 0 | w, c)$ for randomly samples negative pairs. This assumes that a randomly selected context for a given word is likely to result a non-existing pair (w, c) in the corpus C . Using this understanding as a departure point, the problem can be modeled implicitly as a matrix factorization problem for the PPMI matrix. Therefore, a spectral dimensionality reduction algorithm can be casted.

1.3 Neural Networks Methods

In this set of methods, neural networks, with various architectures, are exploited to learn the words vector representations namely, word embeddings, from large training corpora. There have been many efforts that showed success in using neural networks as a mean for extracting word embeddings. Here we discuss some of the most popular methods and review their results.

1.3.1 SENNA embeddings

In their work, Collobert and Weston [8] propose a neural-networks-based method for capturing the word embeddings using an architecture that learns to distinguish between an n-gram sequence from the training corpus and a corrupted version of it. The corrupted sequence is generated by replacing a word in the middle with an arbitrary word from the vocabulary. The replaced word is the word of interest. The training objective is to learn to associate a higher score with the original sequence than the score associated with the corrupted one. The loss is calculated using a hinge function and then backpropagated through the model to change the word embeddings of the word of interest that has been replaced in the corrupted sequence. In their benchmarking results, they showed that the extracted embeddings, when used as feature vectors in the absence of any other features, are fairly well performing in various NLP tasks. They performed their tests using tasks including 1) chunking, 2) part-of-speech-tagging, and 3) named-entity recognition. Figure 1.1 illustrates the basic model they propose for learning the embeddings.

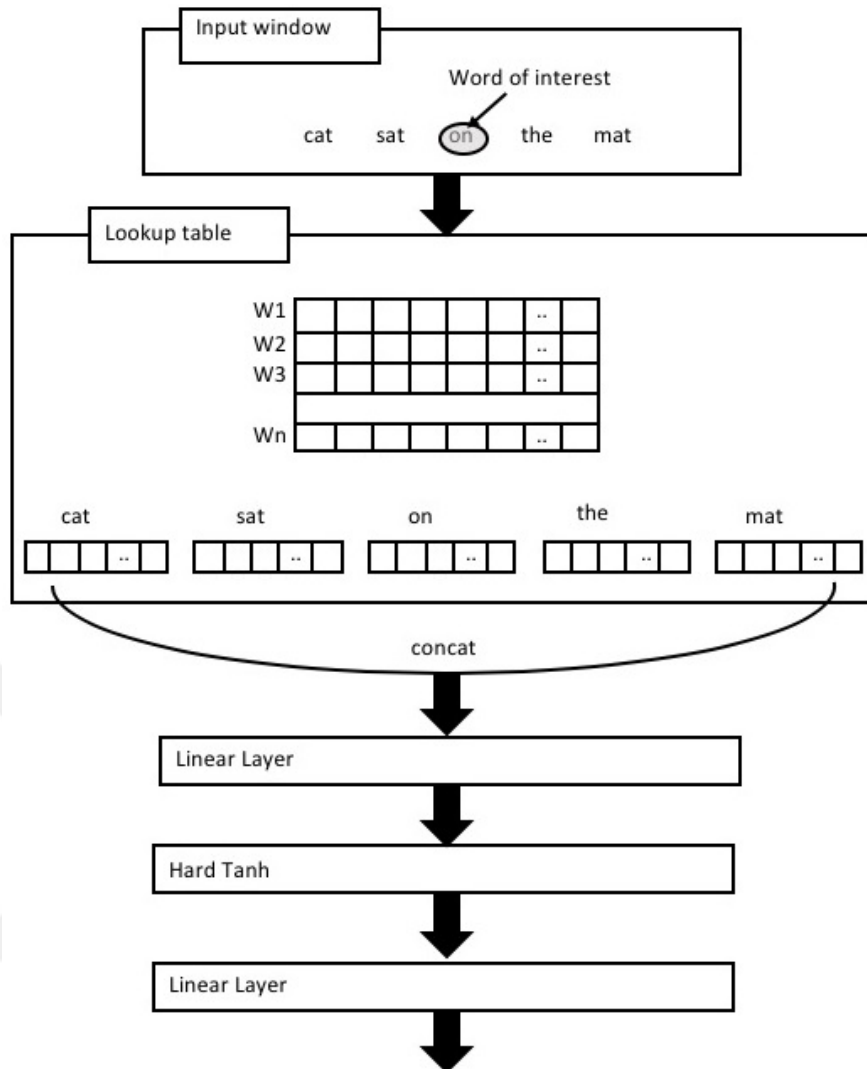


FIGURE 1.1: The neural network architecture proposed by Collobert and Weston [8] for extracting word embeddings using a fixed-sized window. The same architecture was also exploited by Turian et al. [23] with few differences in the experimental settings.

1.3.2 Turian embeddings

In Turian et al. [23], they propose a semi-supervised model for NLP tasks. They implemented the same model described by Collobert and Weston [8] but with the following differences:

- they corrupted the last word in the n-gram sequence instead of the middle.
- they choose different learning rates for the embeddings, i.e. lookup table parameters, and the neural network weights.

In their tests, they not only use the word embeddings, but they also combine them with typical NLP features to improve their results. And for this reason, it is a semi-supervised method, because not all features are learned in an unsupervised manner.

1.3.3 HLBL embeddings

Mnih and Hinton in their work Mnih and Hinton [18] use a variation of Restricted Boltzmann Machines, RBM, called Factored Restricted Boltzmann Machine, FRBM, to obtain the language model. In the model training, they train their model to predict the n th word in an n -gram sequence given all the $n-1$ words. They proposed a log-bilinear loss function to calculate the loss at each iteration. Later, in Mnih and Hinton [19], they introduced a hierarchical log-bilinear model, HLBL, to overcome the drawback of the slow training and testing. This model was primarily inspired by the hierarchical method proposed in Morin and Bengio [20] that prunes the search space for the next word without the need to compute all the probabilities for all the words in the vocabulary.

Before the training of the model, a binary tree for words is constructed. This tree can be constructed using expert data, data-driven methods, or a hybrid method that utilizes the two. Then a hierarchical clustering is performed on this tree based on words usage. The training of the HLBL model is done using the trees resulting from the clustering, and as shown in their results that helped leveraging the performance. However, they use perplexity as a performance measure which is arguably an inadequate metric for evaluating the quality of the information captured by the word embeddings Chen et al. [6].

1.3.4 Huang's embeddings

In their work, Huang et al. [12] try to tackle the challenge of synonymy, i.e. words with multiple meanings. They do this by incorporating the local n -gram contexts with the document global context to disambiguate the multiple meanings of the words. They exploit the same architecture proposed by Turian et al. [23], meaning that, the model is trained to assign a higher score to an n -gram sequence than the score assigned to a corrupted version of it as proposed by Collobert and Weston [8]. However, the corrupted version is the same as the original one after replacing the n th word with an arbitrary word from the vocabulary. This is done on both local and global contexts as shown in Figure 1.2 and scores are combined using a summation function.

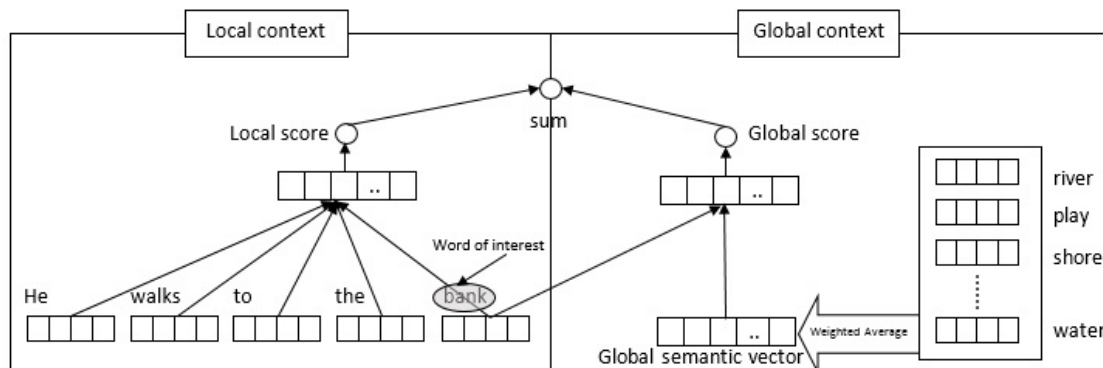


FIGURE 1.2: The architecture proposed by Huang et al. [12] for extracting word embeddings using both local and global contexts to capture the multiple meanings of the same word by learning from the global context i.e. document words. The model uses the same architecture proposed by Turian et al. [23].

1.3.5 Word2vec

One of the most publicly famous available word embeddings are the ones known as, word2vec. The model has been initially proposed by Mikolov et al. [17] and used Skip-gram and Skip-gram with Negative Sampling, SGNS, explained in Section 1.2.1.3 for models training. In their work, they show that the subsampling of the frequent words leads to faster training and better representations of less frequent words. They also propose an approach for learning phrases representations by simply representing each phrase as a single token in the vocabulary. They made the code for training the word and phrase vectors available as an open-source project ¹.

1.3.6 Polyglot

Since most of the methods available focus primarily on English and rich-resources languages, Al-Rfou et al. [1] tried to generate word embeddings for multilingual NLP. In their work, they use a similar architecture that is used in Collobert and Weston [8] to generate SENNA embeddings described in section 1.3.1. However their method differs from the one used to generate SENNA in the following ways:

- they do not limit their models to English, they also train embeddings for a hundred and seventeen other languages.
- they do not do excessive normalization in the preprocessing step to preserve linguistic features that might get lost by doing so. For example, their English model represents the word "Apple" by a vector closer in the representation space to IT

¹code.google.com/p/word2vec

companies vector representations and the word "apple" by a vector closer to fruits vector representations.

They released the embeddings and the code for obtaining them as an open-source project for the community to utilize ².

1.4 Cross Lingual Models

The methods described so far focus on extracting word embeddings one language at a time. Meaning that, the models are trained to induce the distributed representations for just one language and to get the representations for a different language a new model need to be trained. This also means that each language will be represented in a space that is separate from all the other languages. And there is no guarantee that words with the same or similar meanings from different languages will have vectors that are close to each other if we assumed that they are in the same space. However, there is a relatively new research trend that spots the light on how to train the models jointly for two or more different languages. Such that semantically similar words are adjacent to each other in the space irrespective of the language. In the following subsections we review two of the methods that can be utilized to generate jointly trained multilingual word embeddings. One of them Hermann and Blunsom [11] use a data-driven method by utilizing parallel corpora to jointly train the models. While the other Klementiev et al. [13] can be described as an algorithm-driven since it uses the multitasking learning method proposed by Cavallanti et al. [5] to extend the logistic regression algorithm to make it capable of optimizing more than one model jointly.

1.4.1 Klementiev et al. embeddings

In their work, Klementiev et al. [13] aim to extract word embeddings that capture the syntactic and semantic features of the words such that words with similar meanings are brought closer in the space regardless the language. They use a variation of the neural network model described by Bengio et al. [3] and do the joint training using the multitask learning (MTL) setting proposed by Cavallanti et al. [5]. In the following subsections we give more details about the (MLT) and the neural network model they use.

²www.cs.stonybrook.edu/~dsl

1.4.1.1 Multitask Learning

In the multi task set-up, a multitask learner at time t receives a training example for one of the K training tasks. Along with each x_t, y_t pair, a task index i_t is associated. A multitask version of the perceptron algorithm, they propose keeping a weight vector for each task. When a mistake s is made at time t , not only the weights for the task i_t is updated but also for all the remaining $K - 1$ tasks. The update rate for each task is defined by a $K \times K$ matrix called the interaction matrix A . The update at each time step t and task i_t can be written as:

$$v_{j,s} \leftarrow v_{j,s-1} + y_t A_{j,i_t}^{-1} x_t, \forall j \in [1, K] \quad (1.5)$$

1.4.1.2 The Neural Network Model

A neural $n - gram$ language model architecture is used for generating word embeddings for each language. This architecture is similar to the one described by Bengio et al. [3] and illustrated in figure . The model is given as an input a sequence of words $w_{t-n+1:t-1}$ and tries to estimate the probability of the output word $w_t \in V$ where V is the set of all words in the vocabulary. The architecture is used and trained as follows:

- The representations of the input words are obtained and concatenated, preserving the order, to form the input vector $c = (c_{t-n+1}, c_{t-n}, \dots, c_{t-1})$.
- The hidden layer is a simple linear transformation followed by a logistic function to implement the logistic regression algorithm.
- Finally, the output layer is a softmax layer to compute the probabilities of each word in the vocabulary V to be the output word w_t .
- The model is trained using backpropagation using a large corpus.
- The set of free trainable parameters consists of both W , which are the weights of the model layers, and c which are all the representations for all the words in the vocabulary.
- The Learning objective is to maximize the log likelihood expressed as follows:

$$L(\theta) = \sum_{t=1}^T \log P_{\theta}(w_t | w_{t-n+1:t-1}) \quad (1.6)$$

- The training is done using stochastic gradient descent.

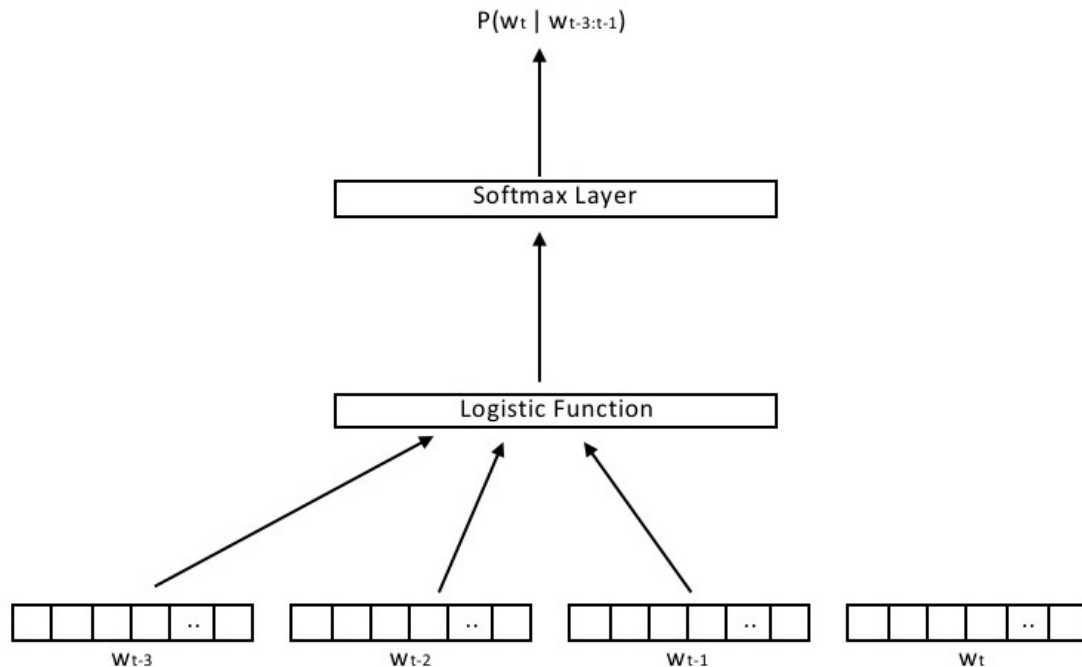


FIGURE 1.3: The architecture proposed by Bengio et al. [3] for extracting word embeddings using n -gram sequences. A similar architecture is used by Klementiev et al. [13] to train multiple models for multiple languages jointly using MTL described in section

1.4.2 Karl Moritz Hermann and Phil Blunsom embeddings

The second cross lingual approach we consider here is proposed by Hermann and Blunsom [11]. They leverage parallel corpora and learn to align the embeddings of semantically equivalent sentences. They also extend their approach to learn representations at the document level as well. Their bilingual model architecture is illustrated in figure and trained as follows:

- For each sentence pair in the parallel corpus, the sentence vector representation of each sentence is composed using a compositional vector model (CVM).
- During the training, the CVM learns how to construct the semantic representation of larger syntactic units given the semantic representations of its parts.
- Assume having two composition functions f and g which map sentences from languages x and y onto distributed representation space. They define the energy of the model given two sentences $(a, b) \in C$ where C is the parallel corpus used for the training and (a, b) are two parallel sentences as follows:

$$E_{bi}(a, b) = \| f(a) - g(b) \|^2 \quad (1.7)$$

- They then try to minimize E_{bi} for all equivalent sentences in the corpus.
- To prevent the model from degenerating, they sample a number of noise sentences n for each pair of parallel sentences (a, b) where n are not equivalent to a with a high probability. Then they define a hinge-loss energy function as follows:

$$E_{hl}(a, b, n) = [m + E_{bi}(a, b) - E_{bi}(a, n)]_+ \quad (1.8)$$

- In their work, they propose two compositional models, the first model is the *ADD* model which simply construct the sentences embeddings by simply summing all the embeddings of the consisting words. This simply can be considered a distributed bag-of-words approach. The second model, is the *BI* model, which captures the bi-gram information using a non-linear *tanh* function over bi-gram pairs in each sentence:

$$f(x) = \sum_{i=1}^n \tanh(x_{i-1} + x_i) \quad (1.9)$$

- Models are trained using backpropagation.

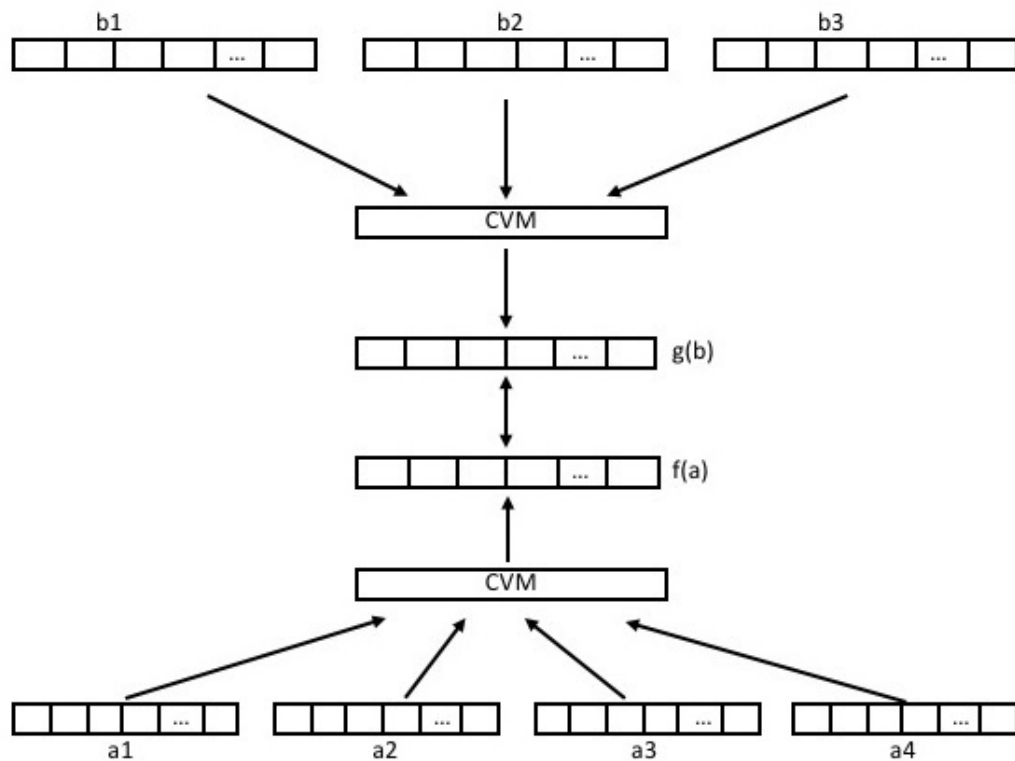


FIGURE 1.4: The bilingual architecture proposed by Hermann and Blunsom [11] for extracting word embeddings leveraging parallel corpora.

1.5 Problem Definition

In the previous sections, we shed a light on distributed word representations, their origin and definition, their usage and methods for extraction, we also reviewed a large portion of what is done so far. However, on one hand, most of the work described in the previous literature focus on high-resources languages as language modelling requires huge corpora. On the other hand, most of the work available consider the word as the basic unit of language, the fact that hinders the performance of most of the methods when dealing with morphologically-rich languages, such as Turkish. Looking closely at the Turkish language, considering the TED Talks English-Turkish parallel corpus ³, the corpus contains around 136700 sentences that split into 246629 words. This is an enormous number compared with the number of the English words in the same 136700 corresponding sentences which is 103928. This, in most of the cases, makes the size of the corpus inadequate to embed sufficient syntactic and semantic information in the word representations. The reason why the number of Turkish words is relatively large is simply that the words can take various forms according to the suffixes entailed to it. And most of the grammar and semantics are represented as suffixes. So in this case, it is a better choice to consider the *morphemes* as the basic language unit instead of the word. To summarize, in the problem of word embeddings extraction from language corpora, two main issues arise. Firstly, the problem of data scarcity in some languages and secondly the fact that not all languages are the same when it comes to the grammar and semantics. This makes the practice of treating all languages the same way quite inadequate. To provide a solution for these problems in order to produce high-quality word embeddings, the available methods for dealing with these pitfalls are explored.

1.5.1 Available methods for tackling the proposed issues

For dealing with low-resources languages, a method that represents multiple language in a joint space can be utilized. In the previous discussion we introduced two methods for representing two or more languages in the same joint space. The first is discussed in section 1.4.1 and can be described as an algorithm-based method since it utilizes an algorithm called multitask learning to simultaneously optimize two models. The second method that can be described as a data-driven method achieves the multilingualism by utilizing sentence-aligned parallel corpora as described in section 1.4.2. One of these methods can be utilized to potentially leverage the quality of the low-resources language word embeddings, by allowing information transfer from high-resources languages to low-resources ones. For example, can consider English and Turkish, using the same setting

³<https://wit3.fbk.eu/>

explained in section 1.4.2, we can start the training with pre-trained embeddings for English and allow the parameters for only Turkish word embeddings to change.

To tackle the issues arise with the morphologically-rich languages, we review some of the techniques from the available literature for integrating compositional morphological representations into language models. In the following subsections, we discuss two of these techniques one simply aggregates the word parts to produce the vector representation for each word. The second method utilizes a recurrent neural network architecture to form the word embeddings.

1.5.1.1 Using Simple Additive Function

One of the methods found interesting, is described in Botha and Blunsom [4] and uses an additive function to aggregate the word embedding vector given the vectors for its consisting morphemes. In their setting, the word embeddings are constructed as follows: This method can also deal with out-of-vocabulary words by constructing them from their building components. For example, if the word *inconvenient* was not encountered during the word embeddings extraction phase, we can still build its vector given the vectors for the morpheme *in* and the word *convenient*. In their setting, the word embeddings are constructed as follows:

$$\overrightarrow{\text{imperfection}} = \overrightarrow{\text{im}} + \overrightarrow{\text{perfect}} + \overrightarrow{\text{ion}} \quad (1.10)$$

They, in order to avoid the order invariance, add the word itself as a component. Meaning that, the embedding for the word *greenhouse* is aggregated as follows:

$$\overrightarrow{\text{greenhouse}} = \overrightarrow{\text{greenhouse}} + \overrightarrow{\text{green}} + \overrightarrow{\text{house}} \quad (1.11)$$

This overcomes the order-invariance of the additive function that would make *handover* = *overhand*.

1.5.1.2 Using Recurrent Neural Network

The second method we discuss here is introduced by Luong et al. [16] and instead of simple additive function it utilizes a simple recurrent neural network architecture to build the word out of its consisting morphemes. Their model as shown in figure 1.5 consists of two layers:

- The morphological RNN layer which generates the word vector by recursively taking its consisting morphemes as input.

- The word-based neural language model which optimizes scores for relevant n-grams using a similar method as the one described in Smith and Eisner [22].

This method, unlike the previously discussed one, takes the order into account which makes it a better choice for the morphologically-rich languages that has prefixes and postfixes as essential components of its syntactic and grammatical rules. However, in the languages which the grammatical and semantic information is mainly represented by only postfixes or prefixes, the additive function is preferred due to the long training time of the RNN and its data-greedy nature.

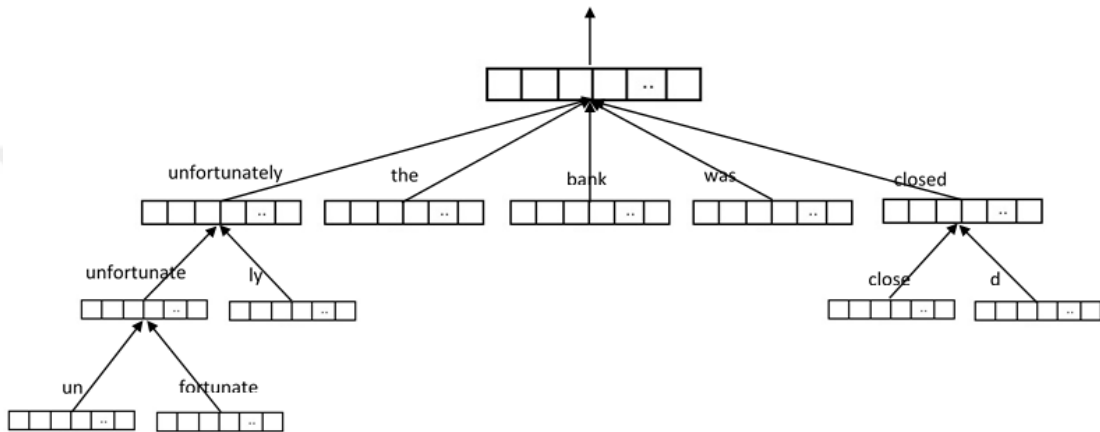


FIGURE 1.5: The layered architecture proposed by Smith and Eisner [22] that utilizes RNN to build word embeddings from morphemes embeddings.

Chapter 2

Proposed Method

2.1 Introduction

In our work, we try to extract multilingual distributed word representations that project all words from different languages in the same vector space while focusing on morphologically rich languages such as Turkish. In our previous discussion, we discussed two methods for extracting multilingual word embeddings. These two models are briefed in section , the first of them can be referred to as an algorithm-based method since it utilizes an algorithm called multitask learning to jointly train two models as described by the original paper Klementiev et al. [13] . The second method that can be described as a data-driven method since it achieves the multilingualism by utilizing sentence-aligned parallel corpora as detailed in Hermann and Blunsom [11]. For overcoming the challenging nature of the morphologically-rich languages, we have previously discussed two methods, one uses a simple additive function to build up the word embedding by adding the vectors for its consisting morphemes as described by Botha and Blunsom [4]. The other method uses a recursive neural network architecture that recursively takes the morphemes consisting a word as an input sequence and outputs the word embedding as described by Luong et al. [16].

2.2 Proposed Framework

In our work, due to its effectiveness and simplicity, we follow the method described in Hermann and Blunsom [11] to extract the word embeddings. Parallel corpora are utilized in order to extract multilingual word embeddings. We used the same model architecture illustrated in figure. In their settings, sentence embeddings are constructed using either

an *ADD* function or *BI* given by the following equation:

$$f(x) = \sum_{i=1}^n \tanh(x_{i-1} + x_i) \quad (2.1)$$

However, in this work, a variation of the equation is used. Instead of considering only bigrams, we consider 6-grams and we apply the tanh function over the vector summation. The equation becomes as follows:

$$f(x) = \sum_{i=1}^k \tanh\left(\sum_{j=i}^{i+6} x_j\right) \quad (2.2)$$

The *ADD* function is used without any modifications.

For Turkish, as a morphologically rich language, a more sophisticated composition function is used. So instead of simply aggregating the sentence embedding given the words, the words need firstly to be constructed using their consisting morphemes. This is done to avoid the pitfalls discussed in the previous chapter represented in the fact that the number of words in the vocabulary would blow up undesirably since the method would consider the word *kedî* which means cat, the word *kediler* which means cats, and the word *kedilerim* which means my cats as three distinct words. This makes the size of the corpus inadequate to embed all the semantic and syntactic information in the words vectors. That is why, the morpheme should be considered as the basic unit instead of the word. To build up the word vector using the morphemes vectors, one of the methods discussed in section can be used. The method introduced by Botha and Blunsom [4] is chosen in favor of the one introduced by Luong et al. [16]. This is due to the insufficient data to train a recurrent neural network architecture. It is also argued that a recurrent architecture would be of a little use in the case of the Turkish language since the semantic, syntactic, and grammatical information are mainly represented in the form of suffixes. Also, the ordering of the postfixes is mainly fixed, for example, the plural suffix comes before the possessive suffix. For this reason, the order of the morphemes is of little importance in the case of Turkish language since it does not change, hence, it does not include any semantic information. That is why, the simple additive method described in Botha and Blunsom [4] is used. However, there is a subtle difference in the way the word vectors are formed here and the way they form them. In their work, they try to avoid the problem of order-invariance of the additive function that would make *handover* = *overhand* by considering the word itself as a component as in the following example:

$$\overrightarrow{\text{greenhouse}} = \overrightarrow{\text{greenhouse}} + \overrightarrow{\text{green}} + \overrightarrow{\text{house}} \quad (2.3)$$

In the case of the Turkish language, and as discussed previously, the order of the morphemes is of little meaning, we do not include the word itself in the equation. This is also useful in avoiding unnecessarily increasing the number of items in the vocabulary. So for the word *arkadaslarim*, we have only three components instead of four. The word is divided into *arkadas*, *lar*, and *im*. In this case the word vector is constructed as follows:

$$\overrightarrow{\text{arkadaslarim}} = \overrightarrow{\text{arkadas}} + \overrightarrow{\text{lar}} + \overrightarrow{\text{im}} \quad (2.4)$$

The framework we use in this work for English-Turkish language pair for example can be illustrated as in figure 2.1.

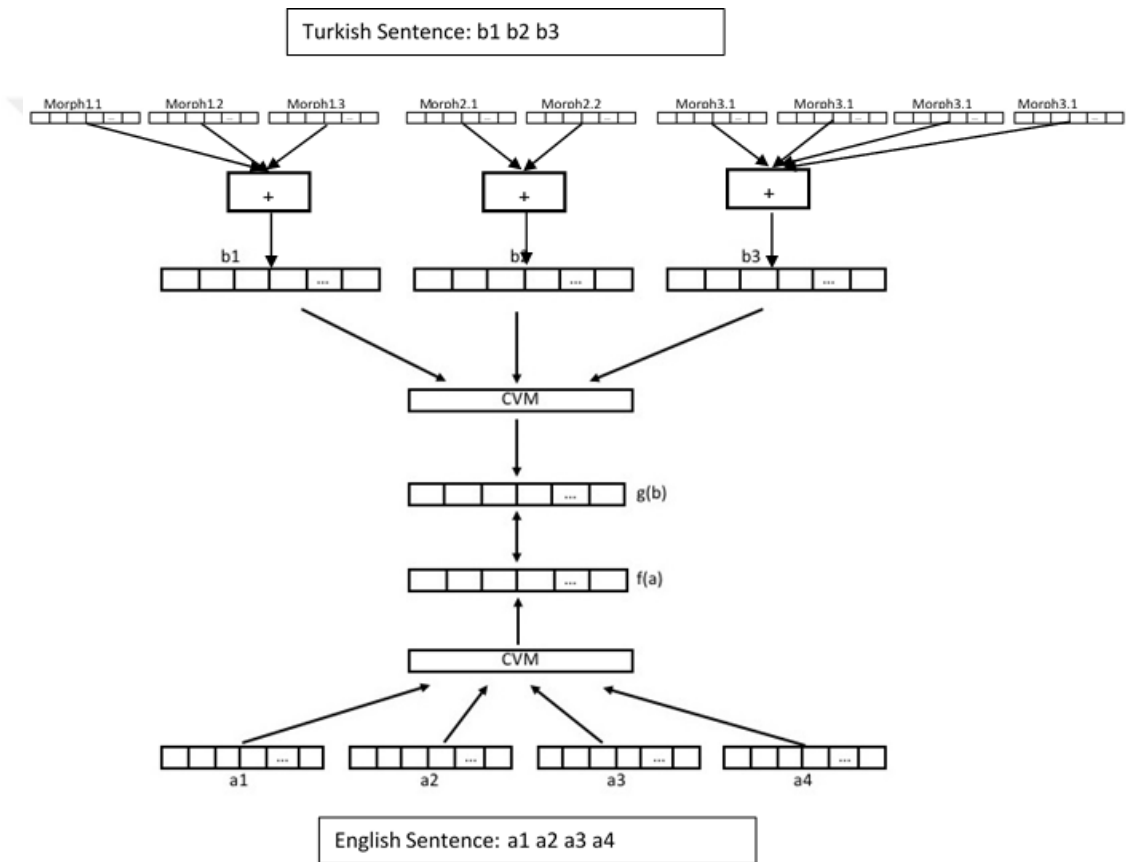


FIGURE 2.1: English-Turkish framework for extracting word embeddings

We use various frameworks for various languages as required by the specific language. The intuition behind our method is that: for each language, we use a composition function to construct the word from its parts, if the language was relatively morphologically rich, and another composition function for constructing the sentences out of their consisting words. For each parallel sentences pair in the corpus, we form the sentence vector for each using the suitable composition functions for forming the words vectors and sentence vectors according to the nature of the language. The composition functions can be anything, as discussed in the previous sections. We vary them choosing between a

simple additive function, n-gram tanh function, or we can even use a recurrent neural network architecture. In the following chapter we conduct various experiments using the illustrated framework and review the results.



Chapter 3

Experiments and Results

3.1 Introduction

In this chapter, we discuss in details how the experiments are performed and how the qualities of the extracted word embeddings using different composition functions are evaluated. We also give details about the datasets and the tools used for the data preprocessing.

3.1.1 Experimental Setting

3.1.1.1 Choosing the composition function

As discussed previously, we can choose the composition functions for composing the words given the words parts and sentences given their consisting words using different schemes. These schemes include, simple additive function, the n-gram tanh function given by the equation 3.1 below, or we can also use a recurrent neural network architecture.

$$f(x) = \sum_{i=1}^k \tanh\left(\sum_{j=i}^{i+n} x_j\right) \quad (3.1)$$

By varying the composition functions various frameworks per language pairs can be achieved. For example, for each language we can achieve $2^3 = 8$ methods to achieve sentence embeddings by varying the word composition function and the sentence composition function. And for each language pair we can achieve $8^2 = 64$ different frameworks.

3.1.1.2 Data preprocessing

For multilingualism, the method described by this work as a data-driven method discussed by [11] is followed. In their work they propose an architecture for extracting the word embeddings utilizing the TED Talks sentence-aligned corpus. This corpus consists of thousands for parallel sentences per language pair. The proposed method aims at minimizing the distance between the vectors that represent the same sentence in the corpus. In the original work, no data preprocessing is done, meaning that, the words are extracted by simply splitting the sentences using the space as a delimiter. In this work, we follow the same architecture but we introduce the idea for not considering the words as the simplest language unit. Instead, we choose between different options, either to consider the words as the basic language block, the tokens, or the morphemes. In the following we define the word, the token, and the morpheme:

- **The word:** words are obtained by simply splitting the sentences using the space as a delimiter. For example, the sentence "I don't know!" would be split to three words: *I*, *don't*, and *know!*. This method ignores the fact that *don't* consists of two words *do* and *not* and the word *know!* is the word *know* and the exclamation mark. To split data into words no special tools are needed, we simply use the string processing libraries included in most of the high-level programming languages.
- **The token:** tokens are obtained by using a tokenizer tool. The one used here is the `nltk`¹ available for python². When a sentence like "I don't know!" is passed to the tokenizer, the tokenizer returns: *I*, *do*, *n't*, *know*, and *!*. This output is language independent, meaning that, if a Turkish sentence was passed, the output would not differ to adapt the language grammar. For example, the sentence "bilmiyorum!" which is the Turkish equivalent to "I don't know!" will be split into: *bilmiyorum*, and the exclamation mark ignoring the fact that the token *bilmiyorum* itself consists of different parts correspond to *I*, *do*, *n't*, and *know* in its English equivalent.
- **The morpheme:** it can be defined as a meaningful morphological unit of a language that cannot be further divided. For example, the word *incoming* can be divided into *in*, *come*, and *ing*. The morphemes can be obtained using a special tool called *morphessor* or *morphologicalanalyzer*. This tool is usually language specific, meaning that, the morphological analyzer for one language cannot be used for another. In the case of our example, the output of a Turkish morphessor for the sentence "bilmiyorum!" which is the Turkish equivalent to "I don't know!" would be: *bil*, *mi* the negation morpheme, *yor* the present continues morpheme, and *um*

¹<http://www.nltk.org/>

²<http://www.python.org/>

the singular first person subject pronoun. The morphological analyzer used for Turkish is the one available as a part of ITU-NLP pipeline ³ that is described by Eryigit [9] and Eryigit et al. [10].

Different methods for data preprocessing can be combined with various frameworks obtained by varying the composition functions. However, the number of possibilities will blow up and it will be unpractical to try them all. That is why only the methods found intuitive are tried knowing the nature of languages and their grammar structures.

3.1.1.3 Training and Parameter Tuning

After designing the architecture for the embeddings extraction, we need to set the training parameters to achieve the best possible quality embeddings. The training objective is to minimize the loss function over the training corpus. The loss function used in this work is the absolute function, meaning that, the model is trained to minimize the absolute distance given by the equation 3.2 between each a,b pair of parallel sentences.

$$E_{bi} = || f(a) - f(b) || \quad (3.2)$$

The models are then trained using backpropagation (backpropagation through time in the case of recurrent architectures) utilizing the output gradients calculated from the loss function in the equation 3.2. For the training we need to tune some parameters we list them below:

- **The learning rate.**
- **The learning rate decay factor**, by which the learning rate is divided after a specific threshold in order to avoid oscillations during the training.
- **The threshold**, the number of epochs after which the learning rate should decay.
- **The maximum number of epochs**, which represents the number of times we should repeat the training using the dataset.
- **The batch size**, the number of sentences per batch.
- And in the case of the n-gram tanh composition function, **the window-size n** is also considered as a design parameter.

³tools.nlp.itu.edu.tr

Due to the long training time of the model at each parameter setting (typically two days) and the large number of possible model architectures, the tuning was done in a trial-and-error manner rather than brute-force search. For parameter tuning, we use a check for sanity called the paraphrase test which we talk about in details in the following subsection.

3.1.1.4 The Paraphrase Test

The paraphrase test is done by simply going through randomly selected 1000 pairs of sentences of the training corpus and forming their embeddings. Then the distances between each two vectors each from one language are computed. For each sentence, if the closest sentence, in terms of embedding, is the one that corresponds to this particular sentence in the parallel corpus, the score increases by 1. Then the total score is reported as a percentages of the correct mapping.

3.1.1.5 The CLDC Test

The cross-lingual document classification or (CLDC) is a problem that was first described by Bel et al. [2] and then was introduced by Klementiev et al. [13] as a method for evaluating the quality of multilingual word embeddings. The (CLDC) task is concerned with training a supervised classifier using documents from one language, the rich-resources language, and testing it using another language, the low-resources one, while maintaining a satisfying performance. The performances of the models are described in terms of the F1-score which is a combined metric that considers both recall and precision of the trained model. In our work, we study how multilingual word embeddings can be used to transfer information from a high-resources language, English, to a low-resources language, Turkish, to successfully train models using English documents and use the same models to classify Turkish documents and still achieving satisfying results. In this test, we train an averaged-perceptron classifier, as suggested by Hermann and Blunsom [11], using the TED-Corpus. This classifier, after trained, should assign one of 14 classes to a document given its embedding. The document embedding is calculated by adding the embeddings of all the consisting words. Since the averaged perceptron is originally a binary classifier, we use one-vs-all to extend it to a multi-class classifier by training 14 binary classifiers each of which is trained to distinguish one class.

3.1.1.6 Hardware and Software

For this work, a TitanX GPU is mainly used due to the high performance that can be achieved by using a GPU for training deep neural networks. Torch7⁴, a scientific computing framework with a wide support for machine learning and neural networks algorithms is used. It is easy to use and computationally efficient since it is built on top of Lua, an easy fast scripting programming language whose compiler is implemented using ANSI-C. Torch7 has support for CUDA which makes it a good choice for implementing and running neural networks on GPU. The code for obtaining the results of this work is open-sourced and made available for the community.

3.1.2 Experimental Results

In our experiments we try different models by varying the data preprocessing method and the composition function. As discussed earlier the number of different combinations is too large to try all the different possibilities. That is why the composition functions and the data preprocessing methods are chosen according to the nature of the language at hand. We firstly try additive functions for all composition functions at both languages using all preprocessing methods discussed in section 3.1.1.2. The second setting tried during the experimentations is achieved by using raw words without any preprocessing at the English side and morphemes at the Turkish side. As discussed earlier, the order of arranging the morphemes in Turkish is fixed (the suffixes are appended to the end in the same order according to their semantic function) which makes the idea of utilizing the simple additive function as a composition function to aggregate the words vectors from their consisting morphemes straightforward. The addition function is preferred over the RNN method due to its simplicity and effectiveness. Also, as mentioned in the previous chapter, the additive method described by Botha and Blunsom [4] is changed subtly for the Turkish language. Instead of adding the word itself as in the following example:

$$\overrightarrow{greenhouse} = \overrightarrow{greenhouse} + \overrightarrow{green} + \overrightarrow{house} \quad (3.3)$$

The word itself is not included since the order of the morphemes in Turkish does not change, to give an example for this, the word *arkadaslarim* can be constructed as follows:

$$\overrightarrow{arkadaslarim} = \overrightarrow{arkadas} + \overrightarrow{lar} + \overrightarrow{im} \quad (3.4)$$

⁴torch.ch

TABLE 3.1: The paraphrase test results for each type of models studied in this work according to the data preprocessing scheme using add function as a composition function.

L1 \ L2	EN-raw	EN-tok
TR-raw	82	
TR-morph		90
TR-tok		68

TABLE 3.2: The paraphrase test results for each type of models studied in this work according to the data preprocessing scheme using n-gram tanh function as a composition function.

L1 \ L2	EN-raw	EN-tok
TR-raw	80	
TR-morph		
TR-tok		50

This helps overcoming the issues that would arise due to including both the words and the morhemes in the vocabulary causing the size of the dictionary to blow up. This would make the size of the training data insufficient for learning meaningful word embeddings.

3.1.2.1 Extracting the Embeddings

The word embeddings for English and Turkish are extracted using one of the models following the proposed framework. Table 3.1 demonstrates the paraphrase test results for each type of models studied in this work according to the data preprocessing scheme. The composition function used at both words and sentences level is a simple additive function. On the other hand, the paraphrase test results illustrated in table 3.2 use the 6-gram tanh function given by the equation 3.1.

3.1.2.2 Evaluating the quality of the embeddings

To evaluate the embeddings quality t-SNE and CLDC tests discussed earlier are used. The t-SNE test results can be illustrated as figures that show the fact that equivalent words project onto the 2D space close to each other. Figures 3.1, 3.2, and 3.3 show the results for t-SNE test for the 6-gram tanh model trained using tokenized data. Figures

3.4, 3.5, and 3.6 show the results for t-SNE test for the 6-gram tanh model trained using raw data. Figures 3.10, 3.11, and 3.12 show the results for t-SNE test for the simple additive model trained using tokenized data. Figures 3.7, 3.8, and 3.9 show the results for t-SNE test for the simple additive model trained using raw data.

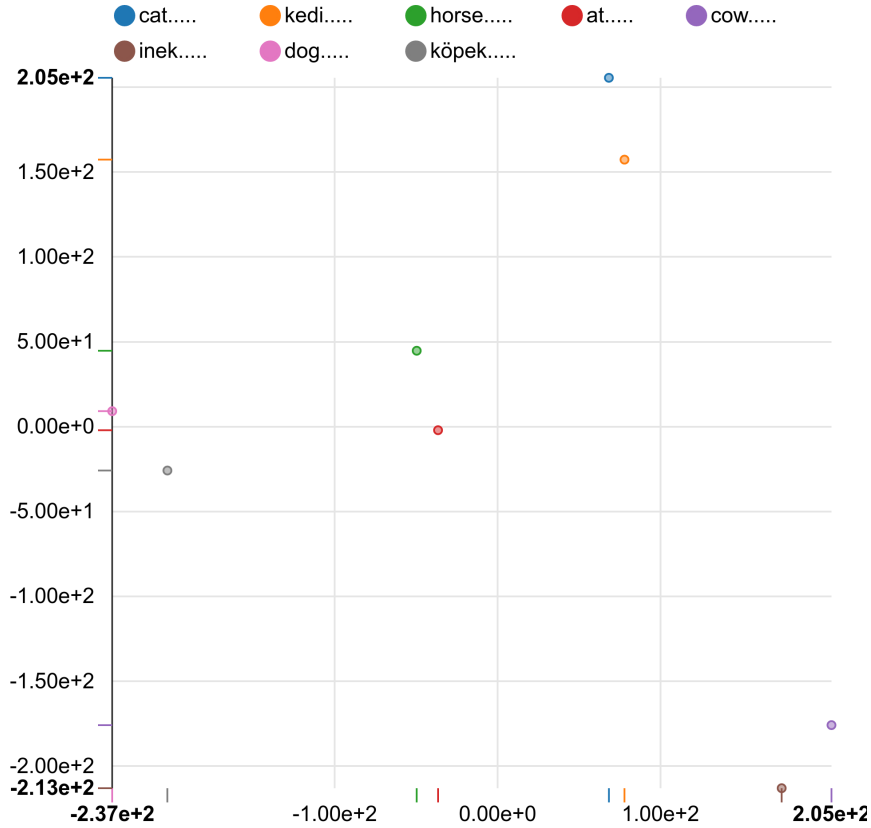


FIGURE 3.1: t-SNE visualization of animals names in Turkish and English using word embeddings extracted by the 6-gram tanh model using tokenization as a preprocessing method

The CLDC results are reported in table 3.3. As we discussed previously, the performance metric used here is the F1-score which is a combined score that takes into account both precision and recall. both tanh-tok and add-tok consider tokenization preprocessing scheme for both languages.

TABLE 3.3: F1-scores obtained by training four models using four types of embeddings. The language of the training set is English while the language for the test set is Turkish.

Embed. set	add-raw	add-tok	tanh-raw	tanh-tok
	0.74	0.73	0.73	0.64

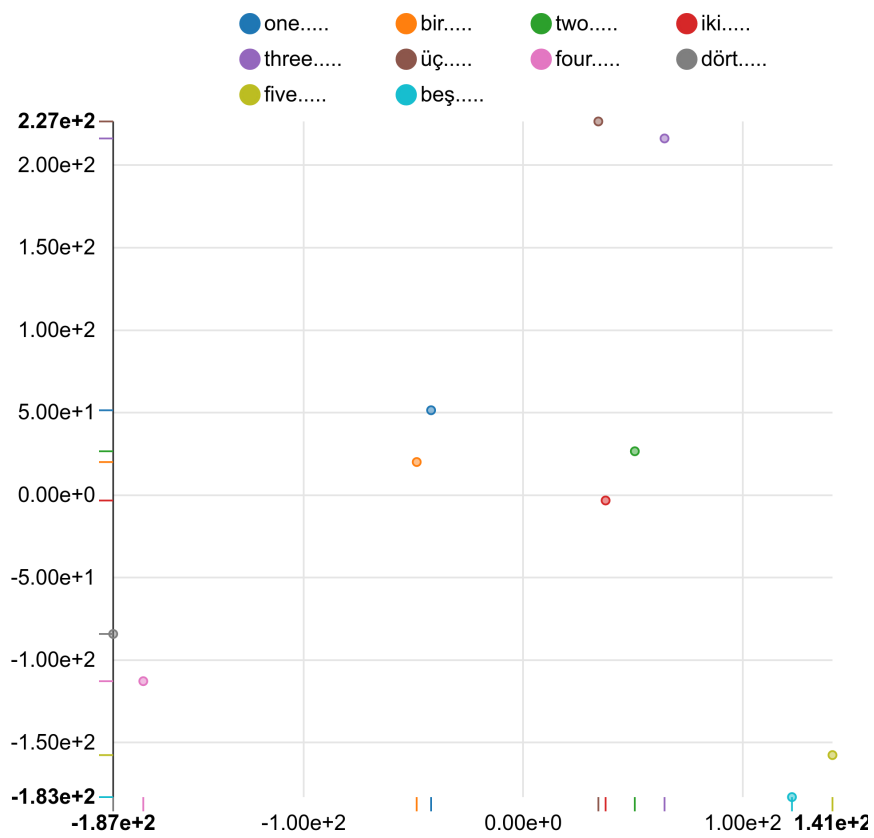


FIGURE 3.2: t-SNE visualization of numbers in Turkish and English using word embeddings extracted by the 6-gram tanh model using tokenization as a preprocessing method

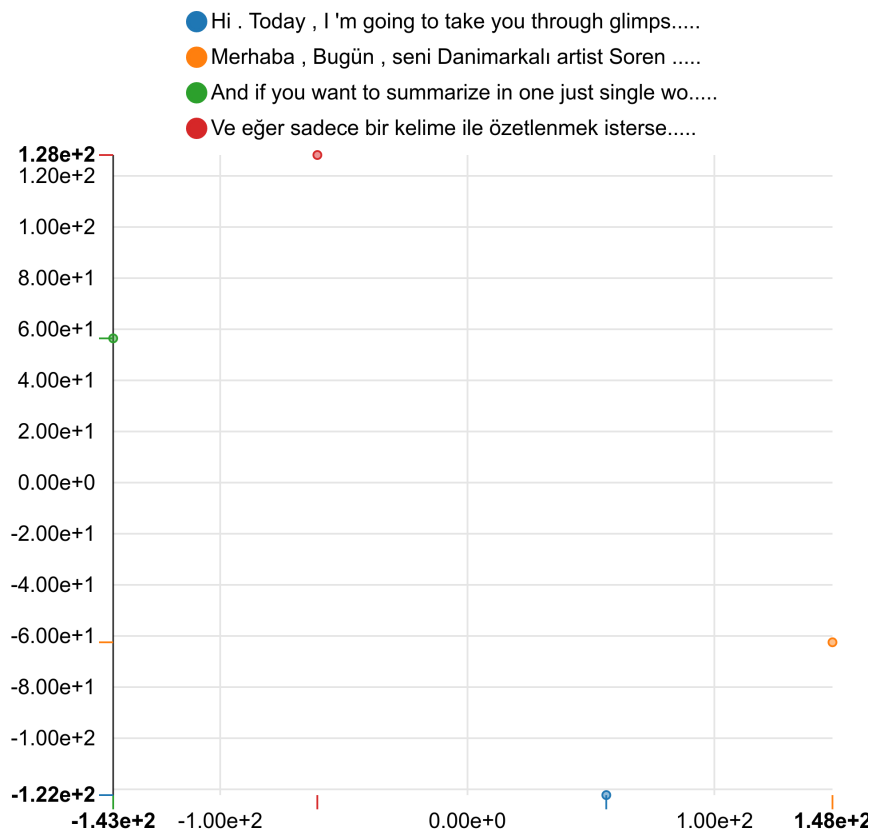


FIGURE 3.3: t-SNE visualization of sentences in Turkish and English using word embeddings extracted by the 6-gram tanh model using tokenization as a preprocessing method

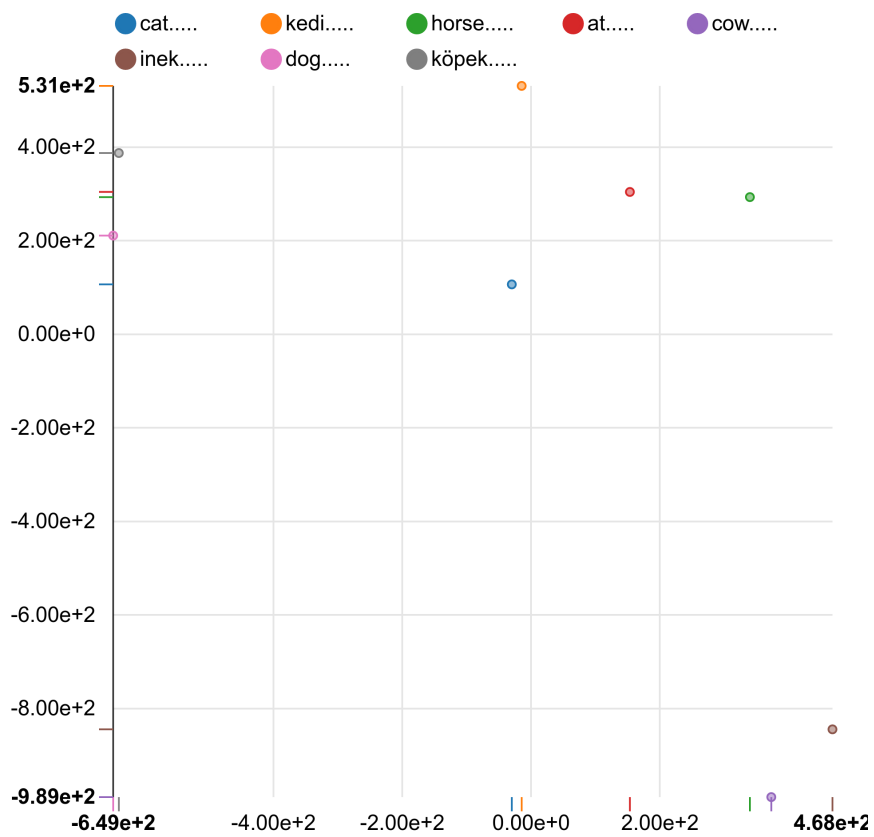


FIGURE 3.4: t-SNE visualization of animals names in Turkish and English using word embeddings extracted by the 6-gram tanh model using raw words

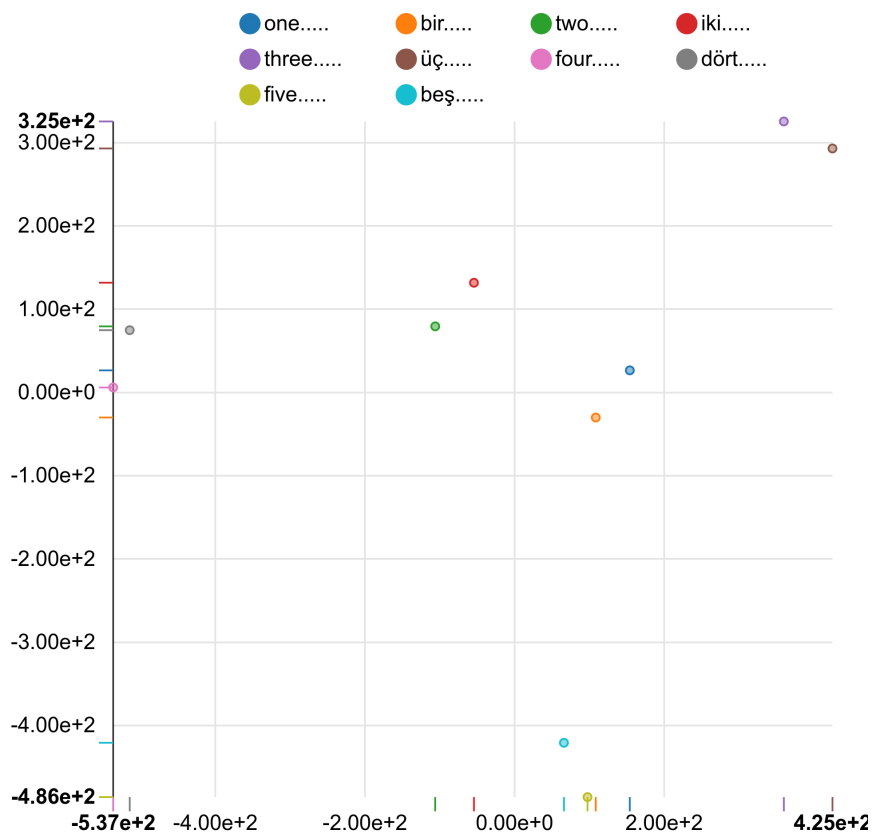


FIGURE 3.5: t-SNE visualization of numbers in Turkish and English using word embeddings extracted by the 6-gram tanh model using raw words

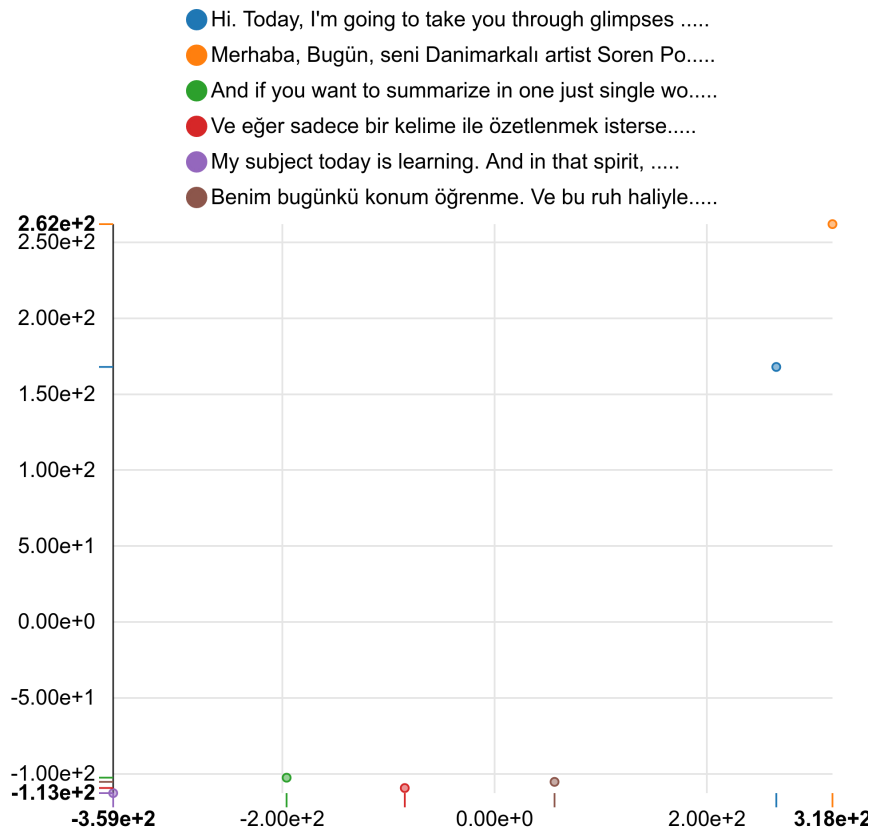


FIGURE 3.6: t-SNE visualization of sentences in Turkish and English using word embeddings extracted by the 6-gram tanh model using raw words

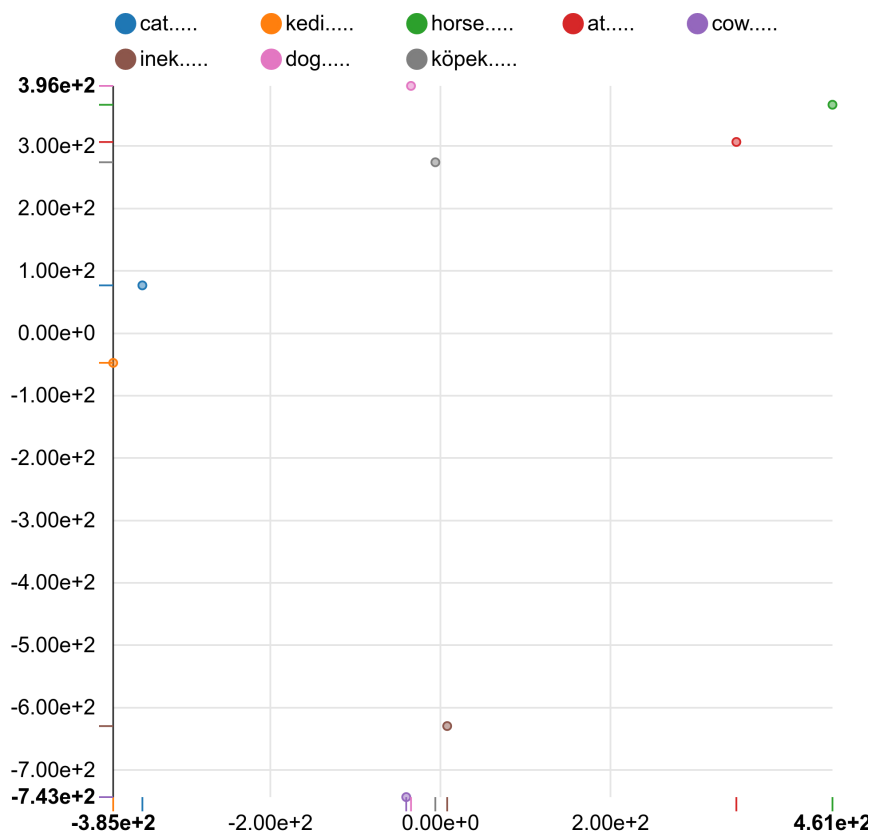


FIGURE 3.7: t-SNE visualization of animals names in Turkish and English using word embeddings extracted by the additive model using raw words

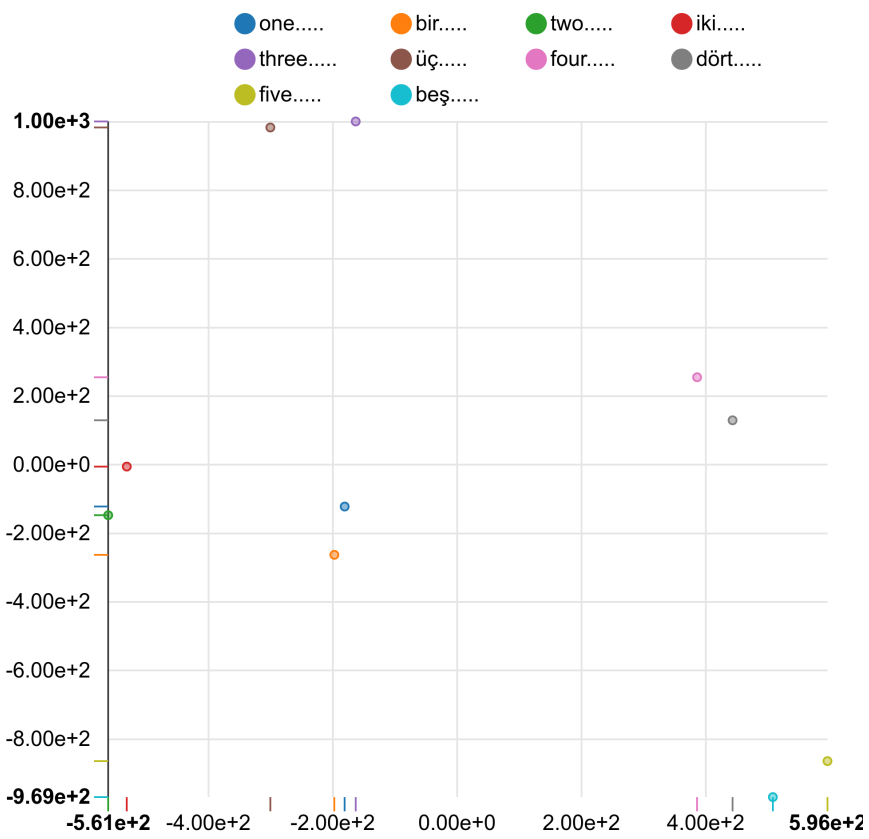


FIGURE 3.8: t-SNE visualization of numbers in Turkish and English using word embeddings extracted by the additive model using raw words

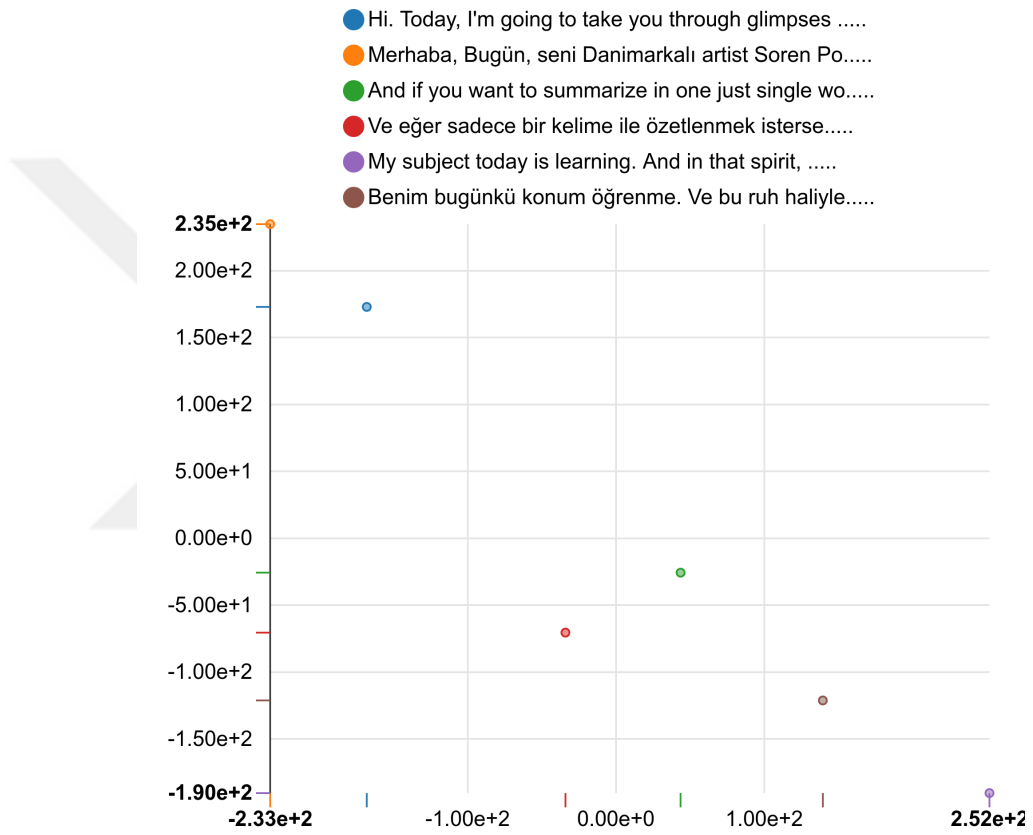


FIGURE 3.9: t-SNE visualization of sentences in Turkish and English using word embeddings extracted by additive model using raw words

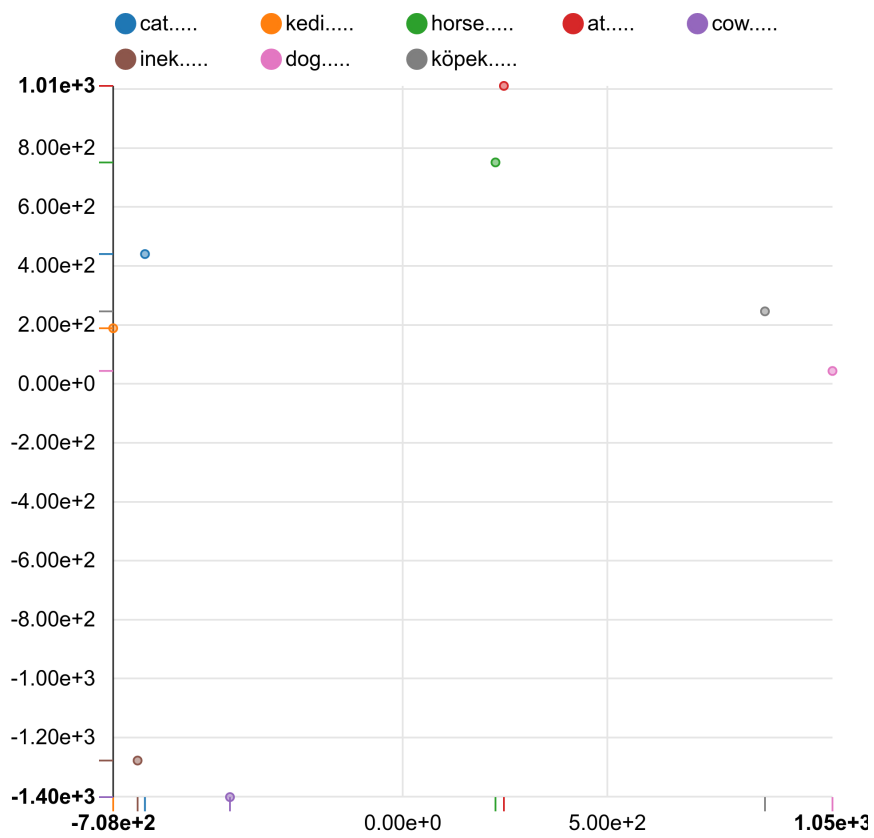


FIGURE 3.10: t-SNE visualization of animals names in Turkish and English using word embeddings extracted by the additive model using tokenization as a preprocessing method

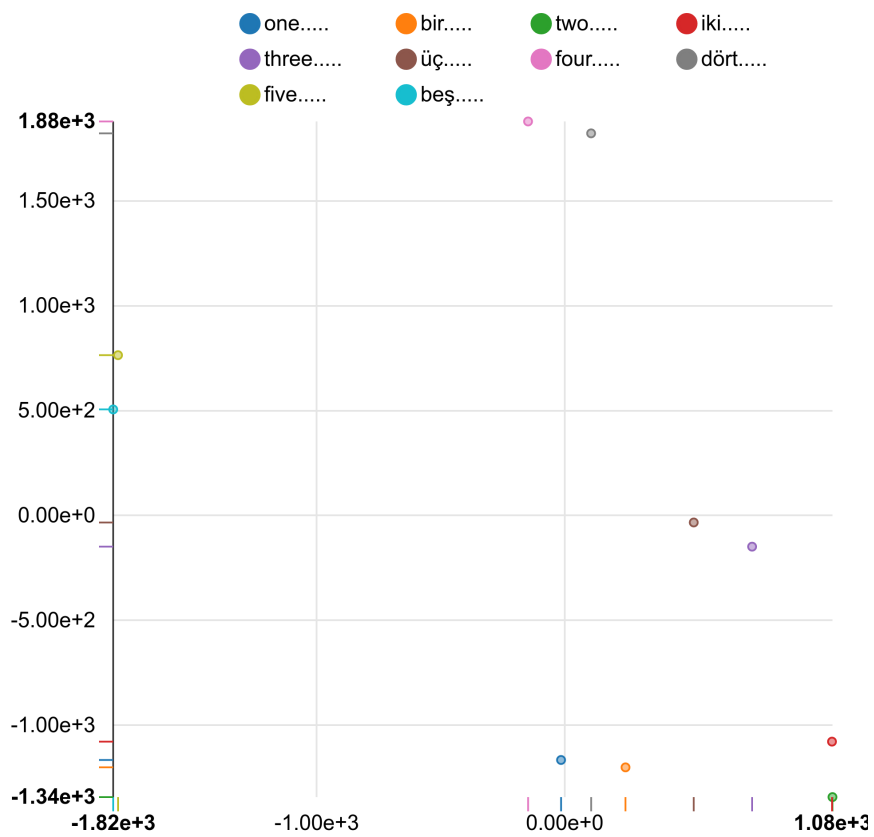


FIGURE 3.11: t-SNE visualization of numbers in Turkish and English using word embeddings extracted by the additive model using tokenization as a preprocessing method

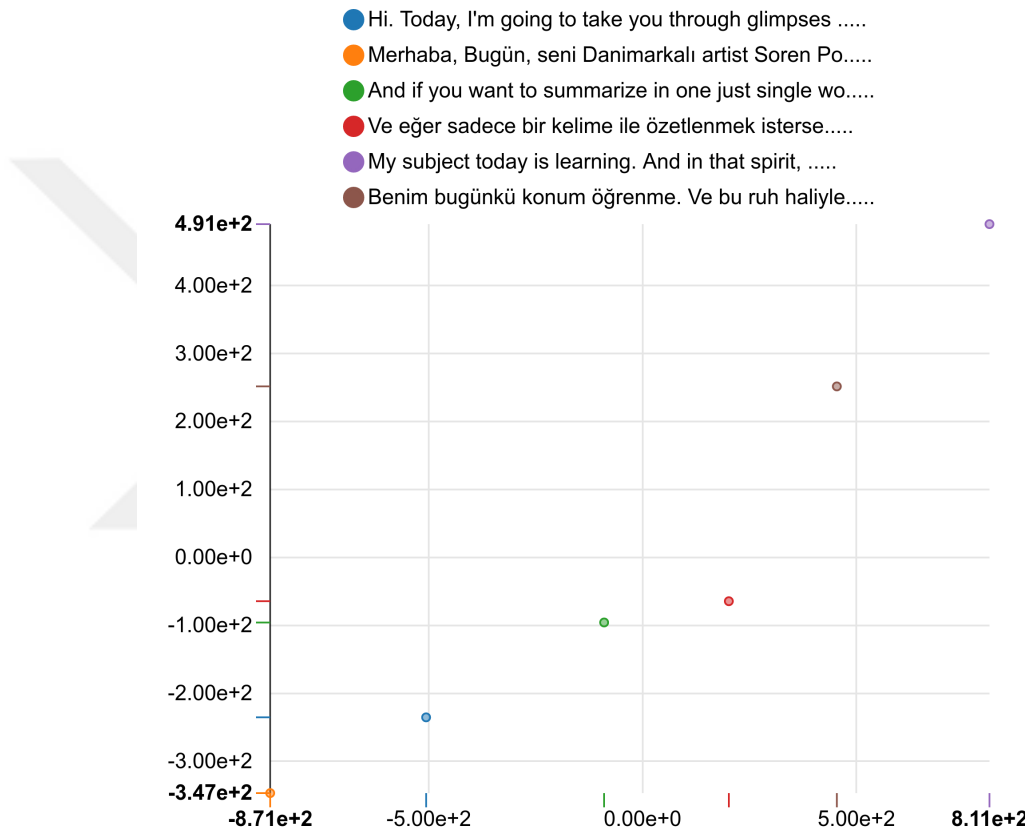


FIGURE 3.12: t-SNE visualization of sentences in Turkish and English using word embeddings extracted by the additive model using tokenization as a preprocessing method

Chapter 4

Conclusion

4.1 Introduction

In this chapter, we overview the methods used for this word, the problem definition, and the main findings.

4.2 Conclusion

In this work, the problem of extracting multilingual distributed word representations is studied. The various method proposed by the literature are also overviewed. As detailed in the background chapter, there are various methods for extracting distributed word representations, also known as, word embeddings by utilizing huge corpora that contain hundreds of thousands of sentences and various language modelling techniques. These word embeddings are numerical vectors that capture the semantic and syntactic information of the words such that similar words should be represented in a proximity in the space. Some of the techniques used for this problem are simply probabilistic methods combined with dimensionality reduction algorithms. Some other techniques are based on training deep neural networks to learn those representations from the available linguistics resources. Most of the work in this field is focusing on learning the word representations for one language at a time. Therefore, the word representations for different languages can be represented as vectors in mutually exclusive spaces. Since the representations would be more meaningful if they enable multilingualism allowing their utilization in a wider spectrum of natural language processing tasks, this word is focused on extracting multilingual word embeddings. For this purpose we use a data-driven method that utilizes neural networks to extract word embeddings for two or more languages from sentence-aligned parallel corpora. This method tries to minimize the distance between

the embeddings for each two parallel sentences. The sentences embeddings are formed by aggregating the embeddings for their consisting words. All embeddings for all words in the vocabulary of a language are stored as weights in the neural model. These words are extracted from the training corpus by splitting all the sentences in the corpus using the space as a delimiter. This method for extracting the words causes many issues to arise with morphologically-rich languages. Considering Turkish as an example for morphologically rich languages, the number of the non-identical words in a corpus will be huge if the same word and its different forms were considered as different words. In this case, more data would be required to learn the representations for such a huge number of words. Since Turkish is a relatively low-resources languages, this would cause the representative power of the model to degrade. To solve this issue, the embeddings for the words are constructed from their *morphemes* before they are aggregated into sentences embeddings. In this work, a hierarchical method that starts at the level of the morphemes embeddings up to the level of sentences embeddings through the word embeddings is proposed. At the transition between each two levels a suitable composition function should be used. In this word, the use of different composition functions that range from simple additive functions to sophisticated recurrent neural networks architectures are discussed. However, for the case study we consider here that includes Turkish and English, the additive function is used at all levels due to its simplicity and effectiveness. The effect of varying the methods for data preprocessing is also studied and compared. Three ways for data preprocessing are considered. The first uses the raw words obtained by splitting the corpus using the space as a delimiter as the basic language unit. The second divides the corpus into a set of tokens. The third method uses a morphological analyzer to generate the morphemes for each word in the corpus and considers the morpheme as the most basic language unit. To evaluate the performance of the model at each settings, three tests were used. The first of them is the paraphrase test, which was used as a primary sanity check to make sure that the model is actually learning something. The paraphrase test was also used to tune the training parameters. The second test, is performed to visualize the embeddings from different languages to show their expressive power. The visualizing method used here is t-SNE, a method that is used to reduce the dimensions of the embeddings for the purpose of visualizing them. The third, and most meaningful, test uses the cross-lingual document classification task (CLDC) to evaluate how well the models embed the multilingual information into the word vectors. In this test, classifiers are trained to assign topic classes to documents. The classifiers are trained using documents from one language and tested using documents from another. For test, F1-score is used as a performance evaluation metric meaning that the model is considered more representative if the F1-score value gets higher. In the experimentations of this work, the model that uses additive functions at all levels for both English and Turkish and uses raw words as the basic language unit for English

and the morphemes as the basic language unit for Turkish scored higher F1-score value than all of the other explored methods.



Bibliography

- [1] R. Al-Rfou, B. Perozzi, and S. Skiena. Polyglot: Distributed word representations for multilingual nlp. *arXiv preprint arXiv:1307.1662*, 2013.
- [2] N. Bel, C. H. Koster, and M. Villegas. Cross-lingual text categorization. In *International Conference on Theory and Practice of Digital Libraries*, pages 126–139. Springer, 2003.
- [3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003.
- [4] J. A. Botha and P. Blunsom. Compositional morphology for word representations and language modelling. In *ICML*, pages 1899–1907, 2014.
- [5] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 11(Oct):2901–2934, 2010.
- [6] Y. Chen, B. Perozzi, R. Al-Rfou, and S. Skiena. The expressive power of word embeddings. *arXiv preprint arXiv:1301.3226*, 2013.
- [7] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.
- [8] 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*, pages 160–167. ACM, 2008.
- [9] G. Eryigit. The impact of automatic morphological analysis & disambiguation on dependency parsing of turkish. In *LREC*, pages 1960–1965, 2012.
- [10] G. Eryigit, J. Nivre, and K. Ofazer. Dependency parsing of turkish. *Computational Linguistics*, 34(3):357–389, 2008.
- [11] K. M. Hermann and P. Blunsom. Multilingual models for compositional distributed semantics. *arXiv preprint arXiv:1404.4641*, 2014.

- [12] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.
- [13] A. Klementiev, I. Titov, and B. Bhattarai. Inducing crosslingual distributed representations of words. 2012.
- [14] R. Lebrecht and R. Collobert. Word emdeddings through hellinger pca. *arXiv preprint arXiv:1312.5542*, 2013.
- [15] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.
- [16] T. Luong, R. Socher, and C. D. Manning. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113, 2013.
- [17] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [18] A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 25th International Conference on Machine Learning*, pages 641–648. ACM, 2007.
- [19] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, pages 1081–1088, 2009.
- [20] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer, 2005.
- [21] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.
- [22] N. A. Smith and J. Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics, 2005.
- [23] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of The Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, 2010.
- [24] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52, 1987.