LEARNING WORD REPRESENTATIONS WITH DEEP NEURAL NETWORKS FOR TURKISH

by

Enes Burak Dündar

B.S., Computer Engineering, Yeditepe University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2019

ACKNOWLEDGEMENTS

I am very grateful to my family for supporting me against all predicaments I have experienced.

I would like to appreciate the guidance of Prof. Dr. Ethem Alpaydın during my research period. I would also like to thank Prof. Dr. Tunga Güngör and Prof. Dr. Olcay Taner Yıldız for accepting to be in my jury.

Also, I appreciate all the professors, research assistants, and students for the intellectual environment in the department.

ABSTRACT

LEARNING WORD REPRESENTATIONS WITH DEEP NEURAL NETWORKS FOR TURKISH

In this study, we analyze the effect of different word embedding methods in representing Turkish texts, namely word2vec, fastText, and ELMo. Word embeddings are used for representing words in a high dimensional vector space such that similar words are placed nearby. This will help in different tasks, such as document classification, machine translation, and so on. We conduct experiments on Turkish corpora of different sizes using word2vec, fastText, and ELMo, and compare them with bag-of-words (BOW). Word2vec works at the word level; fastText works at the character (subword) level and the representation of a word is calculated by combining the representations of subwords. ELMo is context-dependent, that is, the representation of a vector depends on other words in the sentence, whereas word2vec and fastText are context-independent. Learned word embeddings are evaluated on noun and verb inflections, semantic analogy tests, as well as on topic classification of news documents. Our experiments indicate that fastText vectors are better on classification tasks. Word2vec vectors are more useful on semantic analogies.

ÖZET

TÜRKÇE İÇİN DERİN SİNİR AĞLARI İLE SÖZCÜK GÖSTERİMİNİN ÖĞRENİLMESİ

Bu çalışmada, Türkçe metinlerde kullanılan sözcük gösterim yöntemlerinin (word-2vec, fastText ve ELMo) analizine yönelik bir çalışma yapılmıştır. Sözcük gösterimleri, sözcükleri yüksek boyutlu vektör uzayında göstermek için kullanılır. Benzer anlamdaki sözcüklerin bu uzay içinde yakın yerlerde konumlanması amaçlanır. Sözcük vektörleri metin sınıflandırma ve çeviri gibi alanlarda kullanılabilir. Farklı boyutlardaki Türkçe derlemler üzerinde word2vec, fastText ve ELMo yöntemleri üzerinde deneyler yapılıp sözcük çantası yöntemiyle karşılaştırılmıştır. Word2vec yöntemi sözcük seviyesinde çalışırken, fastText harf seviyesindeki gösterimleri kullanarak sözcükleri temsil edebilmektedir. ELMo, cümledeki bağlam bilgisini kullanarak sözcük vektörleri oluşturur. Word2vec ve fastText yöntemleri ise bağlam bilgisini kullanamaz. Öğrenilen sözcük vektörleri sözdizimsel ve anlamsal sınama kümelerinde ve konu sınıflandırmada karşılaştırılmıştır. Deneylerimiz, fastText modelinin konu sınıflandırma konusunda, word2vec modelinin ise anlam benzeşmelerinde daha başarılı olduğunu göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\|D\|$	Number of Dimension
J	Cost value
V	Vocabulary
Q	Set of analogy queries

LIST OF ACRONYMS/ABBREVIATIONS

ANN	Artificial Neural Networks
BiLM	Bidirectional Language Model
BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag of Words
CBOW	Continuous Bag of Word
ELMo	Embeddings from Language Models
GRU	Gated Recurrent Unit
LSTM	Long Short Term Memory
MLP	Multi Layer Perceptrons
MRR	Mean Reciprocal Rank
NLP	Natural Language Processing
NUW	Number of unique words
NW	Number of words
OOV	Out-of-vocabulary
RNN	Recurrent Neural Networks
SGD	Stochastic Gradient Descent

1. INTRODUCTION

There have been lots of studies in the field of natural language processing (NLP) in order to find a good representation of words in a vector space. Such word representations are used as inputs for classification of documents, machine translation, and so on. The quality of word vectors highly affects the success rates on these tasks. For example, Mikolov and his colleagues have introduced a method named word2vec, to obtain distributed word representations [1]. It has been shown that the word2vec model creates better representations than the previous models when tested on various analogy tasks [4]. Word vectors obtained by using the word2vec also have linear relationships and allow "vector algebra": For instance, vec('queen') is the closest one for the result of vec('king')-vec('man')+vec('woman').

In this study, we aim to understand and analyze how word embedding models work on Turkish. Turkish is an agglutinative and morphologically rich language like Hungarian, Finnish, Estonian, and is different from English, the most typically used language in NLP studies.

The methods we used in this thesis are word2vec, fastText and ELMo. Word2vec works at the word level, whereas fastText works at the character (subword) level; the representation of a word is found by combining the representations of subwords. ELMo is different from these in that it is context-dependent; the representation for a word depends on the words that surround it.

In Turkish, nouns and verbs can have a lot of inflections which are determined by looking at the subword information of words, and we evaluate our word vectors on inflections. There are two type of inflections: noun and verb. For instance, the plural form of *kitap*(*book*) is *kitaplar*, not *kitapler*, since vowel harmony is an important factor in Turkish. Also, it is observed that semantic relations are kept in Turkish language models similar to English.

Apart from testing on inflections, we also conduct experiments for classification tasks with Turkish news documents. In the literature, there are a few studies that observe the behavior of word embedding models in Turkish, this thesis is a continuation of those.

This thesis is organized as follows: Word embedding models and background are discussed in Section 2. In Section 3, the corpora and the datasets used in this study are described. In Section 4, experimental results that observe the effect of different word embedding models on different tasks are given. In Section 5, we conclude and discuss possible future research directions.

2. BACKGROUND

2.1. Artificial Neural Networks

2.1.1. Multi Layer Perceptrons

The perceptron is an artificial neural network (ANN) model inspired by the brain [5,6]. Figure 2.1 illustrates a simple perceptron where $x_i \in R^n$ are the inputs, $w_i \in R^{n+1}$ are the weights, and $f()$ is the activation function. The parameter w_0 is called the bias and the output y is calculated as follows:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + w_0\right) \tag{2.1}$$

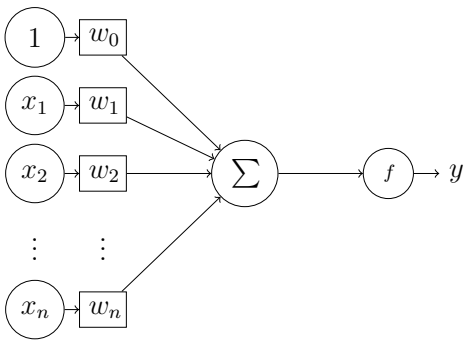

Figure 2.1. The representation of a perceptron in which x_n,w_n, and f respectively denote the input, the weights, and the activation function

Perceptrons can be organized as layers to form a Multi Layer Perceptron (MLP). When the number of parameters in the network increases, both memory and computation requirements increase. The activation function depends on the application. In classification, there are multiple output units y_j each with its own weights and biases. ANN implements a mapping function that transforms the input to its output. In order to accomplish this process, it should learn the weights by reducing the error between the required outputs and the predictions.

There are various ways to compute the error, and they are called loss functions. These functions depend on the task. The backpropagation [7] algorithm uses stochastic gradient-descent (SGD) on the loss function, which uses the derivative of the loss with respect to the weights. If there is more than one layer, then the error is backpropagated to the previous layers.

2.1.2. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a special type of ANNs used for sequential data. It is possible to feed this type of networks with data of varying lengths. While processing sequential data, a RNN model can remember what it has seen before through its recurrent connections. A very special type of RNN is the Long Short Term Memory (LSTM) [8] that contains four gates: input, output, forget, and update. Although LSTM shows good performance on various tasks, there may be the problem called vanishing or exploding gradients [9]. The Gated Recurrent Unit (GRU) [10] is a simpler model compared to the LSTM [11] with two gates: update and reset.

RNN models are applied to different tasks such as machine translation, speech recognition, and so on. A sequence-to-sequence RNN model for machine translation is proposed in [12]. A language model is learned by the RNN model [13]. Such models are used in speech recognition in an end-to-end manner [14].

2.2. Word Embedding Models

In literature, many methods have been proposed to represent words in a high dimensional space. The aim is to put similar words nearby in this space so that this similarity information can be passed on to later tasks such as translation, document classification, and so on. The easiest way to transform words into n-dimensional vectors is the bag of words representation. In recent years, smarter ideas have been proposed, such as word2vec [1], fastText [15], and ELMo [2]. The first two are context-indepentent word embedding models whereas ELMo is context-dependent.

2.2.1. Bag of Words

Bag-of-words (BOW) model is the simplest way of the representation of a document. After creating a vocabulary of words, a document is represented with a binary vector whereas each dimension corresponds to one word. If the word is in the document, then its corresponding dimension in the document vector is set to 1, otherwise to 0.

2.2.2. Word2vec

Word2vec is a well-known method in the field of natural language processing to represent words in a continuous high dimensional space [1]. There are two different architectures: Skip-gram and Continuous Bag of Words (CBOW). In the Skip-gram model, context words in a specified window range are predicted by using the center word. During training, the length of the window is determined in a range determined by user. In Figure 2.2, a sample windowing operation has been shown for different window length. C denotes the center word and Ns are the neighbour words.

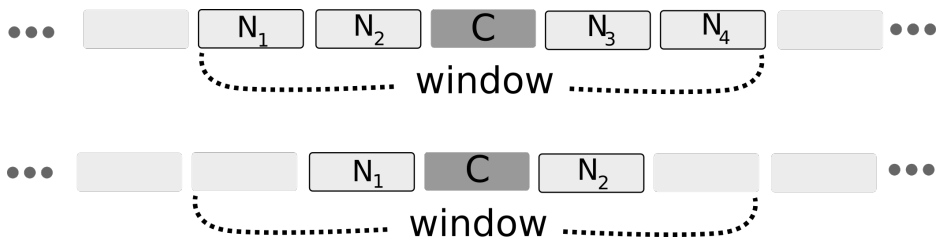

Figure 2.2. Windowing operation on sequential tokens

In the CBOW model, a word is predicted from nearby words in a specified window range. Thus, words that appear frequently in the same context are placed nearby. In Figure 2.3, the Skip-gram model architecture is shown, which is the model we use in this study to learn the word2vec vectors. The important idea is that of transforming an unsupervised learning problem into a supervised learning problem. The context vectors are treated as labels. In training, at first, an embedding layer size of $|V|$x$|D|$ is formed where $|V|$ denotes the number of unique words in the corpus, and $|D|$ denotes the number of dimensions of the new space.

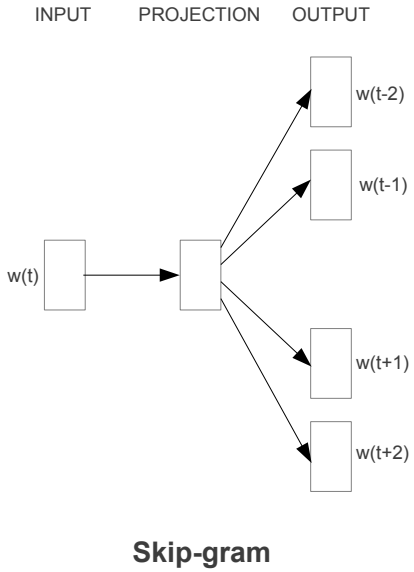

Figure 2.3. Skip-gram model architecture [1]

In the CBOW model, the context word vectors are added up, and it is used as input to the linear classifier. In the Skip-gram model, the current word is used as input of a linear classifier and the context word is the output. Another important contribution is the complexity of the proposed method. Word2vec is a shallow neural network that is capable of learning word vectors from a huge corpus in a reasonable time [1]. When the word2vec model is trained, it is fed with word pairs. Sample training data for selected sequential words from Turkish wikipedia dump are shown in Table 2.1.

When the number of output neurons is huge, it becomes hard to backpropagate the errors. In the word2vec model, we have only one label corresponding to the word which should be predicted. In order to make the learning process more efficient, a negative sampling approach is proposed [16]. In this approach, apart from the label which is called as positive, n negative labels are randomly selected from a non-positive label set. This simplifies computation for small n.

Table 2.1. Sample training data for the sentence:"mimarlık atılımlarının yanı sıra felsefe okulları ortaya çıktı platon"

input	output
felsefe	okulları
felsefe	sıra
felsefe	yanı
felsefe	ortaya
felsefe	atılımlarının
felsefe	mimarlık
felsefe	çıktı
felsefe	platon

2.2.3. FastText

Studies with word2vec have shown promising results on both semantic and syntactic tasks. Yet, even if word2vec is trained on a very large corpus, it cannot find an embedding of a word it has not seen before. It is important to find a vector for such out-of-vocabulary (OOV) words. In FastText [15], the idea is to divide each word into character n-grams. In order to differentiate prefixes and suffixes, a special character (i.e. <, >) is inserted at the beginning and the end of the word. For instance, when n is equal to 3, the character 3-grams of word "this" are the following:

<th, thi, his, is>

Word embeddings of each word and its character grams are considered during the training process. The embedding of a word is calculated by adding up its n-gram vectors. Because the n-grams are the same, one can find an embedding vector for a word not present in the training set. Three to six character grams are enough to achieve reasonable results [15]. The best value of n depends on the language. In this study, we use three and four character grams for Turkish.

For larger n values, there are more character grams, and it is proposed to keep all n-grams in a hashmap so that they can be accessible in a fast way. Similar to word2vec, as the size of the training data increases, the performance of fastText model better [15]. Another advantage of using character n-grams is that the model has the ability to catch syntactic features better. These are discussed in Section 4.

2.2.4. GloVe

GloVe, namely a global vector representation, deals with the word embedding problem by using co-occurence counts of words [3]. While counting co-occurences, the method uses a window with the predetermined range. Then, a 2-dimensional matrix, X, keeping the counts are created where the first dimension denotes the context word of the word in the second dimension. In Table 2.2, an example shows how context words are related to target words. After creating the X, it tries to minimize the following cost equation:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_k + b_i + \tilde{b}_j - log(X_{ij}))^2 \tag{2.2}$$

Table 2.2. Co-occurences probabilities for target words ice and steam [3]

Probability and Ratio	k = solid	k = gas	k = water	k = fashion
P(k|ice)	$1.9\text{x}10^{-4}$	$6.6\text{x}10^{-5}$	$3.0\text{x}10^{-3}$	$1.7\text{x}10^{-5}$
P(k|steam)	$2.2\text{x}10^{-5}$	$7.8\text{x}10^{-4}$	$2.2\text{x}10^{-3}$	$1.8\text{x}10^{-5}$
P(k|ice)/P(k|steam)	8.9	$8.5\text{x}10^{-2}$	1.36	0.96

The weighting function is denoted as $f()$, and is defined in the range between zero and one [3]. The vocabulary size is denoted as V. The model learns w and $\tilde{w}$, the word and the context word vectors in d-dimensions. The authors have denoted that adding up these vectors boosts the performance [3]. While minimizing the cost function, the glove model learns to fit logarithmic values of the co-occurences counts.

2.2.5. ELMo

ELMo [2] is a recent word embedding model with the advantage that it is context-dependent. That is, the vector for a word also depends on the words in its context, whereas with word2vec and fastText, for a word, we always have the same vector regardless of the sentence in which it appears. When compared to word2vec and fasttext, ELMo architecture is much more complex, and requires more time for training and testing.

In ELMo, character embeddings are fed to a bidirectional language model (BiLM) such that it becomes possible to obtain contextualized word embeddings by considering the neighboring words. This is useful because words can have different meanings when they are used in different contexts. Furthermore, it can be used as a feature extractor for different NLP problems. State of the art results are achieved with ELMo on six different tasks [2].

ELMo learns word embeddings as follows: Firstly, tokens of a string are converted to context-independent word embeddings via a character-level Convolutional Neural Network (CNN) followed by two bi-directional LSTM layers. The first LSTM layer takes tokens into account in two directions: from left to right and from right to left. The next LSTM layer on top of the first one uses hidden representations of the previous layer. The goal is to predict the word which is determined as a target at each step. After the model is trained for many epochs on a large dataset, three word embeddings of each token is generated by using the outputs of the three layers: Convolutional and the two LSTM layers. These give representations at different levels of abstraction.

In Figure 2.4, activations of ELMo layers on different tasks are shown after they are softmax normalized. The datasets, SRL, Coref, SNLI, SQUAD, and SST-5, respectively are for semantic role labeling, coreference resolution, textual entailment, question answering, and sentiment analysis. This indicates that which layer is useful depends on the specific task.

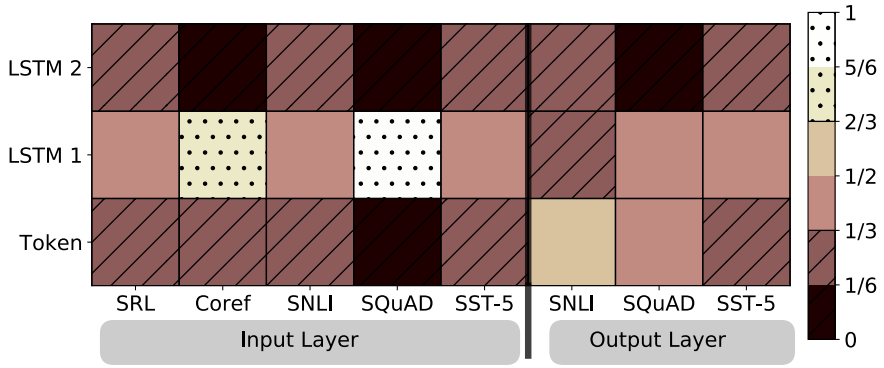

Figure 2.4. Activations of ELMo layers on different tasks [2]

The advantage of ELMo is that it considers the context information when learning a word vector. Therefore, it is possible to extract different vectors for a polysemous word since it has a distinct meaning depending its usage in a sentence. A polysemous example is given in Table 2.3. In GloVe model, the nearest neighbors of the word 'play' are related to game playing. In the ELMo model as we see, the nearest neighbors differ according to the meaning of source. ELMo can also handle out-of-vocabulary words by creating a context-independent vector whereas word2vec and GloVe are not capable of doing this.

Table 2.3. An example for the polysemous word 'play' [2]

Model	Source	Nearest Neighbor
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement.

2.3. Classification

To measure the quality of learned representations, we use them in classification problems. In this study, a linear classifier is used since our aim is to analyze the effect of each vector models. Our task is document classification. That is, the input to the classifier is the representation of the document and the outputs correspond to document classes. During classification process, different type of features are obtained using bag-of-words (BOW), averaging word vectors, and clustering vectors.

2.3.1. Bag-of-words (BOW)

In the BOW model, the most frequent n words that a dataset contains are determined to choose the dictionary. Then, an n-dimensional binary vector is formed in order to represent each document according to the set of words that it contains. If a word is in both bag-of-words and a document, then the vector which represent the document contains 1 for the dimension of that word, and is 0 otherwise. So, the input to the classifier is an n-dimendional binary vector. BOW model is the easiest way to represent a document, and is frequently used as a baseline model.

2.3.2. Averaging word embeddings

When we have vector representations for words, we need to combine them to find a vector for the document. We first use a method like word2vec for words and the average of those word vectors is taken as the vector for the document. The intuition behind this idea is that if documents contain similar words, then the mean vectors will also be similar. So, the input to the classifier will be an m-dimensional vector where m corresponds to the dimensionality of the learned word embedding.

2.3.3. Clustering word embeddings

We also use a clustering-based approach to represent documents. At first, word vector models are clustered by using the k-means algorithm to obtain centroids [17].

Each document is represented by a k-dimensional binary vector. The value of a dimension is set to 1 if the document contains a word whose vector representation is the closest to the mean of that cluster, 0 otherwise.

2.4. Related works on analysis of word embeddings in Turkish

Güngör and Yıldız used the biggest corpus [18] in Turkish to train a word2vec model. Still, the corpus size is small when compared to corpora in English. They conduct experiments using word embedding models on Turkish test sets that are related to morphological properties of the language such as construction and inflection suffixes.

Sen *et al.* used two different sized corpora, *BounWebCorpus* and *Wikipedia Dump*, for training a word2vec language model [19]. Then, word vectors are tested on an analogy test created by the authors. This analogy test contains six semantic and four syntactic questions. In addition to the analogy test, they have also created another test set where the aim is to find a dissimilar word in each group. They also analyze how the dimensionality of word vectors affect success rate.

Ayata *et al.* have done a study for the sentiment analysis of tweets in Turkish [20]. After word vectors are learned, tweet vectors are created by different methods. The first one is to add up vectors of each word in a tweet and averaging them. Another one is that word vectors are multiplied to obtain tweet vectors. After tweet vectors are composed, they are classified by using different models, such as a Support Vector Machine [21] and a Random Forest [22].

Üstün *et al.* propose a new method for creating word embeddings by considering subword information, namely morph2vec [23]. Their model is based on bi-directional LSTMs, and using their abstraction in higher level. They have conducted experiments on both languages: Turkish and English, and showed that their model is better on syntactic tasks.

Güngör *et al.* propose models for handling of the representation of morphologically rich words [24]. They have obtained the state of the art results on the named entity recognition task for Turkish and Czech. They denote that combining three type of embeddings: word, character, and morphological results in better performance.

Ercan and Yıldız have presented a semantic model evaluation dataset for Turkish, namely AnlamVer [25]. It contains word-pairs and their similarity and relatedness scores. While creating the dataset, they consider out-of-vocabulary and rare words for evaluating the word-pairs.

Ertopçu *et al.* has proposed a novel approach for named entity recognition task [26]. They have created a dataset containing 1400 sentences in Turkish for this task. Words in these sentences are annotated such that they can be used for training a classifier. Word labels are learned by operating a context window on sentences.

Topsakal *et al.* applied a shallow parsing on Turkish sentences where the aim is to find a flat non-overlapping parts of a sentence [27]. Windowing operation in [26] is applied again to train a classifier to predict the label of a word. They used six different classifiers by giving word representations as the input to them.

Açıkgöz *et al.* aim to evaluate word sense disambiguation task on Turkish translation of the Penn Treebank Corpus containing 1400 sentences [28]. Apart from word embedding features, they have extracted several features to feed six classifier models. Their findings show that increasing corpus size or vector dimensionality does not help the classification performance [28].

3. CORPORA AND DATASETS

3.1. Corpora

We use the following corpora to train our word representations. Texts in corpora are normalized such that they are lowercased and separated; the exception is apostrophes, because in Turkish, some inflexional suffixes are connected to words with an apostrophe (e.g, "Ahmet'in"). In Figure 3.1, the corpora used in this study are compared in terms of size.

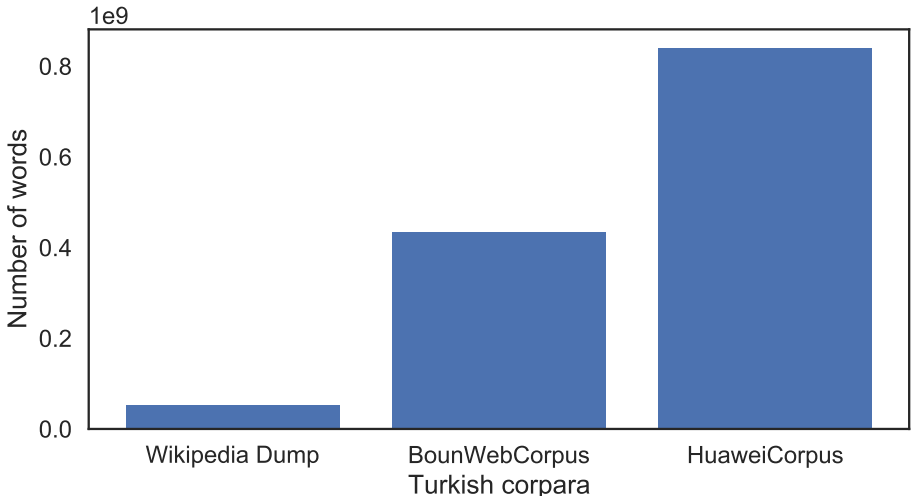

Figure 3.1. Comparison of corpora in terms of the number of words they contain.

3.1.1. Wikipedia Dump

Turkish Wikipedia Dump which we name Wiki, contains wikipedia pages written in Turkish [29]. An extractor tool [30] allows us to extract texts. We concatenate all pages and use it to train our word embedding models. The total number of words is 50,867,627.

3.1.2. BounWebCorpus

Sak et al. have collected news and web pages in order to create a huge corpus which they then preprocessed the corpus containing 433,579,099 words [31]. This corpus is named as Boun.

3.1.3. HuaweiCorpus

Yıldız et al. created the biggest Turkish corpus containing 1 billion words from e-books, websites, and so on [32]. The tokenized version of the corpus is publicly available [18]. After the preprocessing step, the training corpus contains 839,294,373 words. This corpus is named as Huawei. These corpora are used for learning word representation using word embedding models. In Table 3.1, the corpora are compared in terms of the number of unique words. The unique words belonging to the vocabulary are determined for training both word embedding models: word2vec and fastText. As can be seen, when the number of words in a corpus increases, the ratio (NW/NUW) also increases, but even if the corpus size is doubled, the average word appearance is not doubled.

Table 3.1. Comparison of corpora in terms of number of words (NW) and the number of unique words (NUW).

	Wikipedia Dump	BounWebCorpus	HuaweiCorpus
NW	50,867,627	433,579,099	839,294,373
NUW	343,820	934,346	1,439,014
Ratio (NW/NUW)	147	464	583

3.2. Classification Datasets

To evaluate the goodness of various word representation methods, we also use them for document classification. Towards this aim, we use two datasets containing Turkish documents: Hurriyet News and Kemik News.

3.2.1. Hurriyet News

Hurriyet API [33] is used to create a news dataset about three different domains, *Education*, *Sport*, and *Economy*. Each category contains 20,000 news articles. The created dataset is then used for conducting experiments on classification task as reported in Section 4. The detailed statistics about Hurriyet News dataset can be seen in Table 3.2.

Table 3.2. Statistical information about Hurriyet News dataset.

Number of documents	60,000
Maximum document length	1220
Average length of each documents	733.73
Number of unique words	258,815
Maximum length of a word	60
Average length of unique words	9.3854

3.2.2. Kemik News

News articles belonging to various topics are downloaded from the website of *Kemik* which is a natural language processing group in Yıldız Technical University. This dataset has seven categories: economy, arts&culture, magazine, health, politics, sport, and technology, respectively containing 3265, 1155, 2792, 1383, 1849, 9997, and 771 news articles. In Figure 3.2, the distribution of each category can be seen. "Sport" category dominates all other ones. In Table 3.3, detailed information about Kemik News dataset can be found.

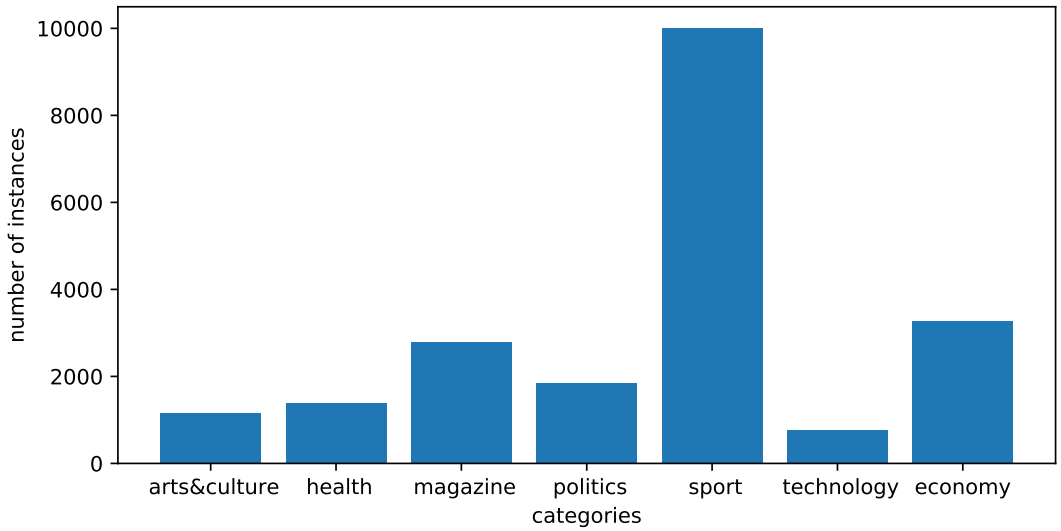

Figure 3.2. Distributions of news on categories.

Table 3.3. Statistical information about Kemik News dataset.

Number of documents	21,212
Maximum document length	964
Average length of each documents	684.34
Number of unique words	129,391
Maximum length of a word	34
Average length of unique words	8.9768

4. EXPERIMENTAL RESULTS

Experiments are conducted on different sized corpora: $TurkishWikipediaDump$, $BounWebCorpus$, and $HuaweiCorpus$, described in the previous section. The original word2vec [4] implemented in C language is used during the experiments. Word vectors are obtained by training word2vec (Skip-gram), fastText, and ELMo models where the configurations of the first two models are set as follows:

While conducting experiments, Keras [34] framework with Tensorflow [35] backend is used. Training is done for three epochs for word2vec and fastText, ten epochs for ELMo. The vector dimensions are set to 100, 200, 300, 400, and 500. The window size is set to five. The number of negative samples for negative sampling is set to five. Finally, the subsampling rate is 0.0004 that is used for balancing the most and less frequent tokens. For fastText, the maximum length of character n-grams is set to four. These configurations are kept the same for all corpora.

Adam [36] optimizer is used in the experiments. It is considered as a combination of AdaGrad [37] and RMSPRop [38] optimizers. Adam extends SGD in that learning rates are specialized for weights instead of using one single learning rate value for the whole network.

Principle Component Analysis (PCA) is utilized in order to reduce high dimensional space to a lower one. Thereby, it becomes possible to represent word vectors in 2-dimensional space. It has been shown that there is a linear relationship between capital cities and countries when high dimensional word embeddings are reduced to 2-dimensions via PCA [16]. In Figure 4.1, we see that we have the same with the word2vec vectors trained on the boun corpus.

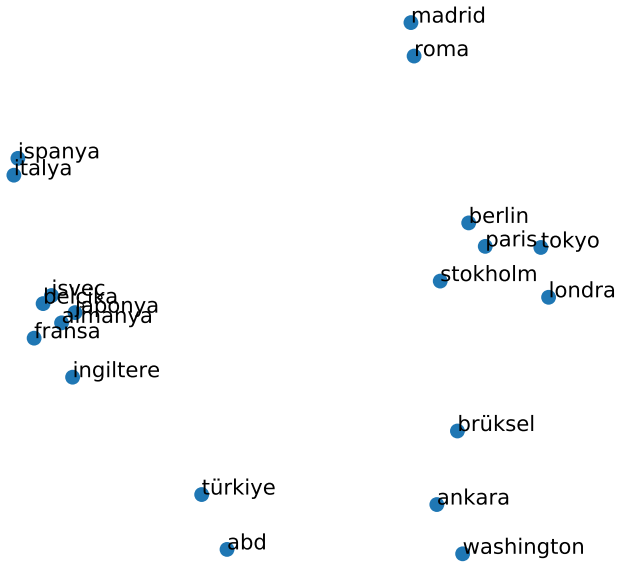

Figure 4.1. 300 dimensional word2vec vectors trained on BounWebCorpus

The original ELMo model architecture is used with a convolutional layer followed by two LSTM layers. The only difference between the original ELMo model and the one used in this study is that the dimensionality of hidden representations is decreased in order to use computing resources efficiently. ELMo is trained with 16M sentences extracted from the *BounWebCorpus*. Our trained models are named using following format: *<first-letter-of-corpus-name>-<vector-dimension>*. The evaluation metric used in this study is the Mean Reciprocal Rank (MRR):

$$MRR\text{-}k = \frac{1}{Q}\sum_{i=0}^{Q}\frac{f(R_i,k)}{R_i} \tag{4.1}$$

Function $f(r,k)$ is equal to 1 if rank r is less than or equal to k, otherwise 0. Vector models are compared according to their performances in each category. After their ranks are determined in each category, the average rank of each model is denoted in *Rank* column of MRR tables. In tables, the best results are denoted with an asterisk.

Table 4.1. Percentages of questions seen in each category of verb inflection tasks

Corpus	olmaz	olurum	olursunuz	olursun	oluruz	olsa	olacak	olsun	olmuş	olmalı	oldu	oluyor	olmakta	Average
w	91.92	25.56	8.79	14.38	46.81	80.44	91.92	53.57	100.0	52.75	100.0	95.24	100.0	66.26
b	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
h	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

4.1. Analogy Tasks

4.1.1. Verb Inflections

In this section, an analogy test [39] is used to evaluate the performance of word vectors obtained with different model configurations. The first column of a table denotes the corpus and the other columns denotes the inflection categories. Each category is represented by using word *ol*.

In Table 4.1, corpora are compared in terms of their ability to cover the questions in an analogy test. This coverage is calculated by considering the existence of words in the vocabulary of each model. It seems that $WikipediaDump$ corpus has a limited coverage and the performance is poor. The two other corpora are larger and contain more information and performance on them is higher.

In Table 4.2, MRR-1 values for verb inflections using word2vec vectors are shown. Wikipedia vectors illustrate better performance when they are shown in a higher dimensional space. Yet, we could not see this behaviour for other vector models. Boun vectors seem to be better than Wikipedia vectors except two categories “olsa” and “oldu”. w-200 model also has the best MRR value for a test on the past tense.

In Table 4.3, MRR-1 values for verb inflections using fastText vectors are shown. The best model on wikipedia data has 300 dimensional vectors. For wikipedia corpus, fastText shows better performance than word2vec except just one category, “oldu”. Moreover, the best models of fastText vectors trained on Boun and Huawei corpora require more dimensionality than word2vec vectors.

In Table 4.4, MRR-5 values for verb inflections using word2vec vectors are represented. The best Wiki vectors are in 300 and 400 dimensions. Yet, they are the worst performing models compared to models trained on other corpora. The best Boun vectors are in 200 and 300 dimensions. In some categories ("olurum", "olacak", "olsun", "oldu", "oluyor"), they have the best MRR-5 scores.

In Table 4.5, MRR-5 values for verb inflections using fastText vectors are represented. Increasing the vector dimensionality in Huawei models results in better MRR-5 values. 100 dimensional vectors of Boun and Huawei corpora are the worst when compared with other models in their groups.

In Table 4.6, MRR-10 values for verb inflections using word2vec vectors are represented. The best model dimensionality is 300 on Boun and Wiki. When the dimensionality is increased, MRR-10 results get better on Huawei data.

In Table 4.7, MRR-10 values for verb inflections using fastText vectors are represented. Again, increasing dimensionality results in a better performance on Huawei data. Wiki and boun vectors in 400 and 500 dimensions are the best in their groups. That means, higher dimensionality of fastText vectors influence MRR-10 values.

In Table 4.8, MRR-1, MRR-5, and MRR-10 values for verb inflections using context-independent ELMo vectors are represented. ELMo vectors are obtained by a context-independent layer. For testing, three different vocabularies are used for creating the vectors. The best results are seen using the wikipedia corpus vocabulary.

In summary, fastText vectors show better performance on verb inflection tasks. When the model dimensionality is increased, the model becomes more successful. Corpus size is also important to reach a better performance on this task.

Table 4.2. MRR-1 values for verb inflections using word2vec vectors; categories are denoted for the Turkish verb *ol*.

Corpus	olmaz	olurum	olursunuz	olursun	oluruz	olsa	olacak	olsun	olmuş	olmalı	oldu	oluyor	olmakta	Rank
w-100	0.159	0.004	*0.000*	0.000*	0.049*	0.161	0.152	0.006	0.340	0.000	0.413	0.299	0.149	3.46
w-200	0.169	*0.008*	*0.000*	0.000*	*0.045*	0.176	0.190	0.006	0.381*	*0.002*	0.476*	0.324	0.167	2.38
w-300	*0.179*	0.012*	*0.000*	0.000*	0.038	0.205	0.198*	*0.008*	*0.357*	*0.002*	*0.468*	0.312	0.167	2.31
w-400	0.172	0.012*	0.011*	0.000*	0.042	0.220*	*0.196*	*0.008*	0.344	*0.002*	0.447	0.333*	*0.178*	*2.00*
w-500	0.184*	*0.008*	0.011*	0.000*	0.040	*0.217*	0.198*	0.011*	0.348	0.005*	0.408	*0.332*	0.184*	1.92*
b-100	0.324	0.217	0.094	0.139*	0.214	0.189*	0.597	0.319	0.509	0.220	0.356	0.691	0.405*	3.69
b-200	0.338	0.265*	0.128*	*0.134*	0.222	*0.187*	*0.625*	0.388 *	0.571*	0.241*	0.445	*0.809*	*0.385*	1.85*
b-300	0.356*	*0.231*	*0.126*	0.127	*0.228*	0.176	0.634*	*0.349*	*0.567*	*0.226*	0.469*	0.815*	0.352	*1.92*
b-400	0.329	0.219	0.115	0.126	0.231*	0.151	0.614	0.323	0.534	0.206	*0.456*	0.801	0.342	3.38
b-500	*0.353*	0.181	0.123	0.119	0.214	0.143	0.610	0.318	0.506	0.211	0.436	0.804	0.325	4.15
h-100	0.383	0.141	0.147	0.166	0.199	0.213	0.318	0.260	0.314	0.200	0.201	0.340	0.302	4.85
h-200	0.473	*0.184*	0.162	0.214	0.265*	0.280*	0.449	0.307*	0.476	0.250*	0.241	0.501	0.442*	2.62
h-300	0.495	0.193*	*0.178*	*0.221*	*0.246*	*0.273*	0.484	*0.302*	0.561	*0.246*	*0.259*	0.547	0.432	2.31*
h-400	*0.498*	0.180	0.177	0.228*	0.231	0.272	*0.520*	0.276	*0.577*	0.240	0.246	*0.575*	0.425	2.69
h-500	0.501*	0.180	0.184*	0.210	0.238	0.258	0.541*	0.255	0.581*	0.226	0.263*	0.604*	*0.440*	*2.38*

Table 4.3. MRR-1 values for verb inflections using fastText vectors; categories are denoted for the Turkish verb *ol*.

Corpus	olmaz	olurum	olursunuz	olursun	oluruz	olsa	olacak	olsun	olmuş	olmalı	oldu	oluyor	olmakta	Rank
w-100	0.491	0.320	0.319	0.213	0.284	0.135	0.516	0.133	0.462	0.254	0.325	0.668	0.711	4.31
w-200	*0.556*	0.447	0.385*	0.250	0.367	*0.142*	0.704	0.171*	0.599*	*0.368*	0.374*	0.727*	0.804	*2.23*
w-300	0.578*	0.557*	*0.352*	*0.301*	*0.390*	0.147*	0.781	*0.159*	*0.592*	0.361	*0.368*	*0.713*	0.834*	1.92*
w-400	0.539	*0.526*	*0.352*	0.316*	0.381	0.133	*0.782*	0.149	0.563	0.365	0.297	0.708	*0.812*	2.69
w-500	0.492	0.478	0.308	*0.301*	0.434*	0.126	0.791*	0.149	0.563	0.375*	0.265	0.706	0.804	3.08
b-100	0.480	0.429	0.414	0.545	0.398	0.150*	0.577	0.304	*0.537*	0.435	0.325	0.679	0.671	4.31
b-200	0.547	0.573	0.609	0.713	0.540	*0.142*	0.695*	0.341*	0.573*	0.557	0.366	0.798	0.813	3.08
b-300	0.595	0.680	0.668	0.744	0.605	0.127	*0.684*	*0.320*	0.525	0.621	0.397*	0.813	0.851	2.69
b-400	0.627*	*0.736*	*0.754*	0.793*	*0.636*	0.109	0.684	0.295	0.520	*0.637*	*0.370*	0.842*	0.867*	2.31*
b-500	*0.622*	0.791*	0.793*	*0.782*	0.661*	0.118	0.645	0.265	0.482	0.658*	0.355	*0.825*	*0.867*	*2.62*
h-100	0.465	0.206	0.278	0.462	0.325	0.135	0.328	0.204	0.312	0.318	0.165	0.336	0.401	4.92
h-200	0.559	0.336	0.448	0.580	0.410	*0.143*	*0.543*	0.255*	*0.455*	0.443	*0.304*	*0.605*	0.607	3.00
h-300	0.588	0.393	0.473	0.609	0.455	0.151*	0.490	0.224	0.408	0.454	0.190	0.488	0.624	2.92
h-400	*0.593*	*0.466*	*0.536*	*0.663*	*0.469*	0.137	0.489	0.219	0.417	*0.496*	0.182	0.477	*0.650*	*2.77*
h-500	0.625*	0.525*	0.627*	0.686*	0.506*	0.110	0.604*	*0.232*	0.490*	0.539*	0.306*	0.704*	0.673*	1.38*

Table 4.4. MRR-5 values for verb inflections using word2vec vectors; categories are denoted for the Turkish verb *ol*.

Corpus	olmaz	olurum	olursunuz	olursun	oluruz	olsa	olacak	olsun	olmuş	olmalı	oldu	oluyor	olmakta	Rank
w-100	0.294	0.024	0.022	*0.000*	0.078*	0.309	0.300	0.019	0.462	0.007	0.618	0.551	0.292	3.77
w-200	0.321	0.020	0.066*	*0.000*	*0.074*	0.340	*0.356*	0.021	0.525	0.008	0.654*	*0.618*	*0.341*	2.62
w-300	*0.340*	*0.036*	0.066*	0.007*	0.068	*0.372*	0.351	0.019	0.541*	*0.010*	*0.640*	0.592	*0.341*	*2.31*
w-400	0.331	0.040*	0.066*	*0.000*	0.072	0.378*	0.362*	*0.024*	0.529	0.008	0.622	0.624*	0.347*	1.92*
w-500	0.353*	0.024	*0.055*	*0.000*	0.064	0.365	0.344	0.030*	*0.531*	0.015*	0.589	0.616	0.347*	2.54
b-100	0.512	0.460	0.250	0.261	0.336	*0.412*	0.758	0.518	0.699	0.425	0.548	0.839	0.577	4.62
b-200	0.596	0.528*	0.309	*0.285*	*0.388*	0.426*	*0.809*	0.605*	*0.765*	0.488*	0.640	0.871*	0.611*	*2.00*
b-300	*0.626*	*0.467*	*0.332*	0.296*	0.386	0.401	0.813*	*0.588*	0.768*	*0.485*	0.671*	0.869	*0.597*	1.92*
b-400	0.610	0.464	0.312	0.296*	0.394*	0.397	0.801	0.571	0.753	0.480	*0.663*	*0.870*	0.593	2.62
b-500	0.645*	0.428	0.343*	0.277	0.371	0.353	0.792	0.566	0.730	0.470	0.641	0.865	0.588	3.54
h-100	0.612	0.282	0.301	0.331	0.359	0.411	0.515	0.423	0.494	0.398	0.377	0.573	0.482	5.00
h-200	0.717	0.371	0.373	0.401	0.439	*0.501*	0.674	0.490	0.667	0.494	0.448	0.732	0.626	3.77
h-300	0.753	*0.389*	*0.389*	*0.429*	*0.460*	0.504*	0.717	*0.492*	0.736	0.501	0.480	0.820	0.657	2.46
h-400	0.769*	0.390*	0.385	0.442*	0.452	0.491	*0.756*	0.493*	*0.752*	0.515*	*0.493*	*0.825*	*0.670*	*1.92*
h-500	*0.762*	0.376	0.406*	0.411	0.470*	0.495	0.774*	0.479	0.771*	*0.511*	0.515*	0.842*	0.697*	1.85*

Table 4.5. MRR-5 values for verb inflections using fastText vectors; categories are denoted for the Turkish verb *ol*.

Corpus	olmaz	olurum	olursunuz	olursun	oluruz	olsa	olacak	olsun	olmuş	olmalı	oldu	oluyor	olmakta	Rank
w100	0.849*	0.600	0.833*	0.000	*0.545*	0.453*	0.868	0.333	0.791	0.418	*0.731*	0.897*	0.891	3.08
w200	0.814	0.668	*0.670*	0.500	0.525	0.373	0.958	*0.343*	*0.819*	0.573	0.727	*0.884*	0.944	3.31
w300	*0.821*	0.783	*0.670*	*0.566*	0.542	0.400	*0.972*	0.352*	0.821*	0.600*	0.743*	0.862	0.962	2.23*
w400	0.784	0.818*	0.659	0.618*	*0.545*	*0.403*	0.983*	0.322	0.803	*0.587*	0.729	0.865	0.968*	*2.38*
w500	0.779	*0.790*	0.560	*0.566*	0.580*	0.391	0.970	0.311	0.793	0.575	0.691	0.877	*0.967*	3.31
b100	0.686	0.672	0.731	0.804	0.560	0.448	0.790	0.583	0.762	0.664	0.515	0.826	0.849	5.00
b200	0.778	0.832	0.881	0.901	0.677	*0.497*	0.881	0.679*	0.840*	0.766	*0.585*	0.884	0.932	3.23
b300	0.806	0.900	0.916	0.939	0.712	0.463	*0.902*	*0.645*	0.809	0.808	0.579	0.900	0.944	2.92
b400	0.829*	*0.937*	0.958*	0.970*	*0.715*	0.506*	0.922*	0.632	*0.826*	0.848*	0.607*	0.917*	*0.955*	1.46*
b500	*0.826*	0.953*	*0.953*	*0.960*	0.722*	0.475	0.896	0.601	0.798	*0.840*	0.575	*0.906*	0.958*	*2.38*
h100	0.656	0.409	0.518	0.717	0.471	0.325	0.493	0.433	0.496	0.538	0.309	0.547	0.586	5.00
h200	0.774	0.600	0.700	0.808	0.564	0.394	*0.748*	*0.555*	*0.705*	0.723	*0.480*	*0.799*	0.775	3.23
h300	0.787	0.647	0.740	0.849	0.593	0.462*	0.735	0.553	0.682	0.736	0.371	0.740	0.803	3.15
h400	*0.794*	*0.682*	*0.781*	*0.874*	*0.607*	0.431	0.747	0.559*	0.701	*0.767*	0.356	0.758	*0.824*	*2.38*
h500	0.808*	0.725*	0.855*	0.903*	0.642*	*0.458*	0.833*	0.554	0.782*	0.808*	0.503*	0.855*	0.844*	1.23*

Table 4.6. MRR-10 values for verb inflections using word2vec vectors; categories are denoted for the Turkish verb *ol*.

Corpus	olmaz	olurum	olursunuz	olursun	oluruz	olsa	olacak	olsun	olmuş	olmalı	oldu	oluyor	olmakta	Rank
w-100	0.377	0.032	0.044	*0.000*	0.085	0.393	0.362	0.022	0.531	0.010	0.696	0.637	0.361	4.38
w-200	*0.408*	0.040	0.088	0.007*	0.093*	0.409	0.414*	0.029	*0.597*	0.012	0.733*	0.702*	0.417	2.46
w-300	*0.408*	0.055*	*0.110*	0.007*	*0.087*	*0.432*	*0.410*	*0.036*	0.600*	*0.019*	*0.720*	0.684	*0.423*	1.92*
w-400	0.399	0.043	0.099	*0.000*	*0.087*	0.443*	0.401	0.032	0.594	0.013	0.714	*0.700*	0.425*	2.46
w-500	0.418*	*0.051*	0.121*	*0.000*	0.085	0.417	0.399	0.038*	0.588	0.027*	0.677	0.687	0.425*	*2.38*
b-100	0.622	0.523	0.345	0.336	0.395	0.531	0.802	0.591	0.761	0.500	0.616	0.858	0.649	4.69
b-200	0.698	0.603*	0.430	0.355	*0.456*	0.541*	0.841*	0.675*	0.834*	0.583	0.695	0.886*	0.703*	*2.15*
b-300	*0.742*	*0.584*	*0.435*	0.376*	0.453	*0.536*	*0.838*	*0.674*	0.834*	*0.598*	*0.713*	*0.880*	0.693	2.00*
b-400	0.735	0.577	0.422	*0.371*	0.465*	0.506	0.823	0.667	*0.824*	0.611*	0.719*	0.879	*0.696*	2.54
b-500	0.761*	0.539	0.445*	0.356	0.450	0.500	0.814	0.669	0.809	0.596	0.709	0.879	0.685	3.15
h-100	0.674	0.328	0.380	0.400	0.431	0.485	0.588	0.483	0.557	0.473	0.435	0.655	0.536	4.92
h-200	0.776	0.456	0.454	0.485	0.512	0.574	0.739	0.554	0.723	0.578	0.528	0.807	0.691	3.92
h-300	0.815	0.468	0.489	0.514	0.535	*0.587*	0.786	0.560	0.783	0.585	0.565	0.854	0.725	2.92
h-400	0.838*	*0.473*	*0.504*	0.538*	*0.546*	0.574	*0.804*	*0.566*	*0.798*	*0.611*	*0.576*	*0.857*	*0.739*	*1.92*
h-500	*0.829*	0.477*	0.516*	*0.516*	0.567*	0.589*	0.823*	0.577*	0.813*	0.612*	0.607*	0.865*	0.756*	1.15*

Table 4.7. MRR-10 values for verb inflections using fastText vectors; categories are denoted for the Turkish verb *ol*.

Corpus	olmaz	olurum	olursunuz	olursun	oluruz	olsa	olacak	olsun	olmuş	olmalı	oldu	oluyor	olmakta	Rank
w-100	0.814	0.648	0.714	0.426	0.481	0.408	0.887	0.386	0.758	0.508	0.715	0.894	0.907	5.00
w-200	0.864	0.779	*0.846*	0.640	0.568	0.478	0.982	*0.454*	*0.888*	0.672	0.831	*0.916*	0.971	3.31
w-300	*0.898*	0.873	0.868*	0.698	0.593	0.526	0.991	0.460*	0.891*	0.686	0.837*	0.902	0.983	2.38
w-400	0.902*	0.901*	0.791	0.772*	*0.606*	0.538*	0.996*	0.417	0.882	*0.692*	0.821	0.915	*0.985*	2.08*
w-500	0.890	*0.897*	0.758	*0.757*	0.674*	*0.534*	*0.992*	0.410	0.876	0.696*	*0.833*	0.928*	0.987*	*2.23*
b-100	0.744	0.760	0.811	0.860	0.615	0.581	0.838	0.676	0.816	0.712	0.596	0.877	0.883	5.00
b-200	0.832	0.886	0.935	0.950	0.708	0.624	0.909	0.770*	0.890*	0.814	0.668	0.907	0.947	3.46
b-300	0.860	0.942	0.954	0.967	*0.747*	0.632	*0.943*	*0.747*	0.878	0.854	0.661	0.919	0.955	2.85
b-400	0.876*	*0.968*	*0.978*	0.987*	0.746	0.667*	0.958*	0.742	*0.886*	0.887*	0.702*	0.930*	*0.961*	1.62*
b-500	*0.870*	0.977*	0.985*	*0.978*	0.758*	*0.645*	0.925	0.730	0.861	*0.885*	*0.683*	*0.927*	0.964*	*2.08*
h-100	0.713	0.507	0.632	0.793	0.513	0.406	0.566	0.521	0.561	0.605	0.368	0.654	0.630	5.00
h-200	0.821	0.679	0.776	0.860	0.622	0.509	0.810	*0.661*	*0.791*	0.786	*0.556*	*0.851*	0.834	3.31
h-300	0.832	0.708	0.817	0.897	0.646	*0.574*	0.807	0.655	0.773	0.808	0.467	0.808	0.862	3.15
h-400	0.847*	*0.733*	*0.850*	*0.922*	*0.664*	0.571	*0.830*	0.651	0.783	*0.827*	0.441	0.834	*0.876*	*2.46*
h-500	*0.843*	0.778*	0.890*	0.939*	0.694*	0.600*	0.890*	0.673*	0.849*	0.867*	0.594*	0.913*	0.901*	1.08*

Table 4.8. MRR values of context independent ELMo vectors for verb inflections where inflection categories are denoted by the an example of the Turkish verb *ol*.

Corpus	olmaz	olurum	olursunuz	olursun	oluruz	olsa	olacak	olsun	olmuş	olmalı	oldu	oluyor	olmakta	Rank
w-mrr1	0.278*	0.198*	0.176*	0.265*	0.176*	0.146*	0.338*	0.075*	0.057*	0.059*	0.066*	0.183*	0.407*	1.00*
b-mrr1	0.214	0.130	*0.122*	*0.130*	*0.111*	*0.105*	0.243	*0.048*	*0.032*	0.029	*0.059*	0.109	0.357	2.46
h-mrr1	*0.215*	*0.134*	0.117	0.124	0.104	0.101	*0.258*	0.048	0.027	*0.031*	0.048	*0.142*	*0.366*	2.54
w-mrr5	0.519*	0.312*	0.242*	0.346*	0.269*	0.353*	0.512*	0.138*	0.127*	0.101*	0.122*	0.308*	0.541*	1.00
b-mrr5	*0.399*	0.197	*0.195*	*0.221*	*0.168*	*0.258*	*0.400*	0.097	*0.088*	*0.055*	0.097	0.242	0.489	2.38
h-mrr5	0.398	*0.202*	0.193	*0.221*	0.165	0.253	0.396	*0.098*	0.086	0.053	*0.100*	*0.244*	*0.491*	2.54
w-mrr10	0.586*	0.360*	0.275*	0.412*	0.322*	0.423*	0.570*	0.173*	0.152*	0.119*	0.153*	0.350*	0.578*	1.00*
b-mrr10	*0.472*	0.224	0.213	*0.242*	*0.206*	*0.325*	*0.472*	*0.123*	0.115	*0.067*	0.120	*0.295*	*0.534*	2.31
h-mrr10	0.466	*0.227*	*0.215*	*0.242*	0.197	0.320	0.469	0.120	*0.116*	0.065	*0.122*	0.289	0.529	2.62

4.1.2. Noun Inflections

Similar to previous subsection, noun inflections are evaluated by using an analogy test set [39]. Table 4.9 shows how well questions in the test set is covered by our vector models trained on Wiki, Boun, and Huawei.

Table 4.9. Percentages of questions seen in each category.

Corpus	konular	konumuz	konum	konunuz	konun	konuları	konusu	konudan	konuyu	konuya	günce	konunun	konuyla	konuda	Average
w	95.92	27.64	35.99	2.59	66.33	95.92	100.0	100.0	100.0	100.0	38.04	100.0	91.76	87.77	74.42
b	100.0	100.0	100.0	100.0	95.92	100.0	100.0	100.0	100.0	100.0	76.09	100.0	100.0	100.0	98.00
h	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

In Table 4.10, MRR-1 values for noun inflections using word2vec vectors are shown. The best performing model has 200 dimensional word vectors trained on Boun-WebCorpus. Another important result is that models with 200 and 300 dimensional vectors have high quality word representations. For the category “günce”, all models show poor performances.

In Table 4.11, MRR-1 values for noun inflections trained using fastText vectors are shown. Higher dimensional word vectors get better MRR results. On Wiki, 400 dimensional vectors show the best performance on ranking. For the category “günce”, performances of all fastText models are generally better than word2vec ones .

In Table 4.12, MRR-5 values for noun inflections using word2vec vectors are shown. The best models of Wiki, Boun and Huawei use 200, 300, and 500 dimensional vectors respectively.

In Table 4.13, MRR-5 values for noun inflections using fastText vectors are shown. The best Wiki and Huawei models are in 500 dimensional space, and the ranking variance between other vector models is high. For the category “konular”, high dimensional vectors are more successful to represent singular and plural nouns.

In Table 4.14, MRR-10 values for noun inflections using word2vec vectors are shown. The best models of Wiki, Boun and Huawei use 300, 400, and 500 dimensional vectors. For the category “günce”, all models show poor performances.

In Table 4.15, MRR-10 values for noun inflections using fastText vectors are shown. The best models of wiki, boun and huawei use 500, 400, and 500 dimensional vectors. For the category “günce”, fastText models are better than word2vec ones.

In Table 4.16, MRR-1, MRR-5, and MRR-10 values for noun inflections using context independent ELMo vectors are shown. ELMo vectors are obtained by a context-independent layer. For testing, three different vocabularies are used for creating the vectors. The best results are seen with ELMo on the Wiki corpus vocabulary. ELMo vectors using the Boun vocabulary is the second best, and the last one is on the Huawei vocabulary. ELMo vectors based on Wiki vocabulary is the best except on the category “konunuz”.

In summary, fastText models seem to be better than word2vec ones. The possible reason of that is the capability of the fastText architure on subword information. Also, vectors in higher dimensionality show better performance.

Table 4.10. MRR-1 values of word2vec vectors for noun inflections in which inflection categories are denoted by the example of Turkish nouns “konu” and “gün”.

Corpus	konular	konumuz	konum	konunuz	konun	konuları	konusu	konudan	konuyu	konuya	günce	konunun	konuyla	konuda	Average
w-100	0.129	0.003	0.030	0.000*	0.258*	*0.062*	*0.225*	0.058*	*0.184*	0.135*	0.000*	*0.288*	*0.037*	*0.044*	2.07*
w-200	0.128	0.009	0.027	0.000*	0.247	0.066*	0.259*	0.052	0.189*	*0.123*	0.000*	0.275	0.041*	0.039	*2.43*
w-300	0.135	0.019*	0.049*	0.000*	0.246	0.059	0.219	0.050	0.175	0.116	0.000*	0.290*	0.031	0.050*	*2.43*
w-400	0.137*	*0.015*	*0.039*	0.000*	*0.253*	0.060	0.192	*0.053*	0.161	0.116	0.000*	0.281	*0.037*	0.028	2.57
w-500	*0.137*	*0.015*	0.032	0.000*	0.231	0.048	0.179	0.052	0.140	0.122	0.000*	0.274	*0.037*	0.031	3.29
b-100	0.230	0.146*	*0.098*	0.096*	0.210	0.083*	0.257	0.118*	0.280	0.190*	0.024*	0.378	0.097	0.072*	2.57
b-200	*0.238*	*0.127*	0.111*	*0.084*	*0.224*	*0.054*	0.294*	*0.108*	0.306*	*0.140*	0.014	0.406*	0.115*	*0.060*	1.71*
b-300	0.253*	0.092	0.085	0.077	0.231*	0.047	*0.292*	0.076	*0.304*	0.125	0.014	*0.391*	*0.098*	0.039	*2.50*
b-400	*0.238*	0.065	0.093	0.060	0.223	0.037	0.267	0.065	0.294	0.105	*0.019*	0.381	0.075	0.036	3.43
b-500	0.232	0.061	0.081	0.056	0.218	0.021	0.260	0.047	0.293	0.090	0.009	0.382	0.063	0.026	4.43
h-100	0.187	0.150*	0.053	*0.108*	0.174	0.043	*0.215*	0.104	*0.221*	0.100	0.014*	0.267	0.085	*0.048*	3.14
h-200	0.213	*0.149*	*0.064*	0.104	0.194	*0.043*	0.219*	0.146*	0.230*	0.094	0.014*	0.318	0.104	0.051*	*2.36*
h-300	*0.214*	0.134	0.085*	0.109*	0.213	0.044*	0.183	*0.139*	0.214	0.102	*0.011*	0.333	0.113*	0.043	2.21*
h-400	0.222*	0.118	*0.064*	0.095	*0.220*	0.038	0.148	0.122	0.202	*0.107*	0.007	0.341*	*0.107*	0.039	2.93
h-500	0.212	0.108	0.048	0.076	0.225*	0.035	0.160	0.102	0.201	0.108*	0.007	*0.340*	0.098	0.027	3.79

Table 4.11. MRR-1 values of fastText vectors for noun inflections in which inflection categories are denoted by the example of Turkish nouns “konu” and “gün”.

Corpus	konular	konumuz	konum	konunuz	konun	konuları	konusu	konudan	konuyu	konuya	günce	konunun	konuyla	konuda	Average
w100	0.306	0.169	0.081	0.464*	0.359	0.310	0.288	0.389	0.264	0.299	0.057	0.452	0.220	0.194	4.50
w200	0.445	*0.265*	0.148	0.464*	*0.413*	0.460	0.334*	0.460	*0.312*	0.382*	0.095	0.520	0.338	*0.280*	2.71
w300	0.503	0.292*	0.160	0.464*	0.409	0.524	*0.324*	0.482	0.311	0.360	*0.143*	0.530	0.399*	0.260	*2.50*
w400	0.506*	0.259	*0.168*	0.464*	0.424*	*0.536*	0.313	0.516*	0.314*	*0.375*	0.133	0.553*	*0.359*	*0.280*	1.79*
w500	*0.504*	0.246	0.197*	*0.321*	0.409	0.561*	0.270	*0.514*	0.298	0.374	0.152*	*0.533*	0.334	0.305*	*2.50*
b100	0.282	0.320	0.187	0.484	0.205	*0.103*	*0.230*	0.180	*0.287*	0.157*	*0.067*	0.375	0.207	0.105	3.57
b200	0.330	0.358	*0.220*	*0.549*	0.235	0.095	0.258*	0.216	0.304*	*0.130*	0.052	0.390*	0.260	0.121	2.71
b300	0.350*	0.372*	0.223*	0.540	*0.236*	0.088	0.219	*0.232*	0.267	0.113	0.043	*0.386*	*0.287*	0.140*	*2.50*
b400	*0.336*	*0.370*	0.209	*0.549*	0.239*	0.099	0.171	0.246*	0.246	0.117	0.071*	0.390*	0.290*	*0.136*	2.21*
b500	0.317	0.308	0.214	0.555*	0.219	0.105*	0.162	0.225	0.230	0.117	0.062	0.360	0.278	0.130	3.36
h100	0.189	0.214	0.143	0.332	0.187	0.062	0.209	0.154	0.236	0.111	0.022	0.265	0.161	0.067	4.64
h200	*0.299*	0.299	*0.158*	0.429	0.299*	0.077	0.266*	0.211	0.292*	0.159*	*0.025*	0.455*	0.222	*0.099*	*2.29*
h300	0.247	0.333*	0.190*	0.500*	0.261	0.101*	0.223	*0.217*	0.247	0.129	0.022	0.377	0.245	0.096	2.57
h400	0.264	0.309	0.150	*0.483*	0.266	0.089	0.198	0.241*	0.238	0.126	0.029*	0.384	0.255*	0.095	2.86
h500	0.322*	*0.329*	0.131	0.500*	*0.284*	*0.090*	*0.241*	0.216	*0.272*	*0.134*	*0.025*	*0.425*	*0.250*	0.124*	2.07*

Table 4.12. MRR-5 values of word2vec vectors for noun inflections in which inflection categories are denoted by the example of Turkish nouns “konu” and “gün”.

Corpus	konular	konumuz	konum	konunuz	konun	konuları	konusu	konudan	konuyu	konuya	günce	konunun	konuyla	konuda	Average
w-100	0.311	0.019	0.064	*0.000*	0.395	0.231	0.402	0.184	0.382	0.304	*0.000*	0.464	0.113	0.126	4.00
w-200	0.346	0.037*	0.079	0.036*	0.408*	0.260*	0.500*	0.181	0.455*	*0.323*	*0.000*	0.511*	0.138*	*0.142*	1.71*
w-300	0.355*	0.037*	0.089*	0.036*	0.408*	0.232	*0.485*	*0.191*	0.448	0.330*	*0.000*	*0.507*	0.122	0.162*	*1.79*
w-400	*0.348*	0.031	0.089*	0.036*	*0.401*	*0.239*	0.477	*0.191*	*0.450*	0.307	0.009*	0.486	*0.126*	*0.142*	2.14
w-500	0.325	*0.034*	*0.081*	*0.000*	0.381	0.206	0.440	0.198*	0.430	0.316	0.009*	0.475	0.123	0.130	3.00
b-100	0.379	0.285	0.207	0.203*	0.343	0.255	0.438	*0.268*	0.485	0.364*	*0.029*	0.583	0.252	0.167	3.50
b-200	0.448	0.300*	0.241*	*0.191*	0.362	*0.283*	0.487	0.286*	0.521	0.346	*0.029*	0.610	0.324*	0.177*	*2.50*
b-300	0.471*	*0.292*	0.214	0.186	*0.382*	0.289*	0.520	0.248	0.542	0.347	*0.029*	0.623	*0.304*	*0.170*	2.43*
b-400	*0.468*	0.276	*0.235*	0.174	0.384*	0.246	*0.541*	0.251	*0.559*	0.344	*0.029*	0.627*	0.278	0.166	2.79
b-500	0.458	0.270	0.214	0.166	0.380	0.211	0.547*	0.225	0.576*	*0.348*	0.033*	*0.625*	0.260	0.152	3.21
h-100	0.316	0.293	0.137	0.220	0.274	0.163	0.360	0.241	0.389	0.232	0.018*	0.439	0.205	0.134	4.36
h-200	0.367	0.304*	0.169*	0.239*	0.300	0.207	*0.384*	0.314*	0.412	0.273	*0.014*	0.509	0.247	0.148	2.71
h-300	0.372	*0.300*	0.169*	*0.231*	0.336*	0.198	0.372	*0.301*	0.429	0.305	*0.014*	*0.542*	*0.269*	0.175	*2.43*
h-400	*0.379*	0.298	*0.166*	0.210	*0.325*	*0.210*	0.375	0.294	*0.435*	*0.327*	*0.014*	0.533	0.268	*0.181*	2.50
h-500	0.410*	0.279	0.143	0.197	0.336*	0.219*	0.419*	0.282	0.463*	0.349*	*0.014*	0.548*	0.273*	0.183*	2.00*

Table 4.13. MRR-5 values of fastText vectors for noun inflections in which inflection categories are denoted by the example of Turkish nouns “konu” and “gün”.

Corpus	konular	konumuz	konum	konunuz	konun	konuları	konusu	konudan	konuyu	konuya	günce	konunun	konuyla	konuda	Average
w-100	0.511	0.400	0.000	0.000	0.617	0.560	0.568	0.600	0.537	0.528	0.036	0.756	0.501	0.391	4.86
w-200	0.658	0.452	0.355	*0.536*	0.609	0.710	0.598	0.767	0.556	0.616	0.410	0.783	0.648	0.568	3.71
w-300	*0.712*	*0.455*	0.370	0.571*	0.619	0.743	*0.622*	*0.809*	0.575	0.610	0.362	0.787	0.727*	0.630	2.57
w-400	0.707	0.446	*0.377*	*0.536*	0.632*	*0.749*	0.653*	0.812*	*0.615*	0.651*	*0.419*	*0.799*	0.701	*0.642*	*2.00*
w-500	0.732*	0.486*	0.404*	0.500	*0.622*	0.762*	0.621	0.808	0.618*	*0.633*	0.486*	0.803*	*0.708*	0.662*	1.64*
b-100	0.437	0.575	0.428	0.723	0.355	0.338	0.435	0.390	0.483	0.336*	0.133	0.593	0.425	0.262	4.07
b-200	0.516	0.672	0.465	0.779	0.401	0.331	0.465*	0.457	0.500*	*0.329*	0.167*	*0.627*	0.556	0.286	*2.79*
b-300	*0.534*	*0.684*	0.480*	*0.785*	0.412*	0.366	*0.449*	*0.474*	*0.497*	0.315	0.133	0.640*	0.601	0.307	2.14*
b-400	0.535*	0.720*	*0.479*	0.804*	*0.403*	*0.388*	0.420	0.503*	0.480	0.314	*0.152*	0.626	0.633*	*0.333*	2.14*
b-500	0.535*	0.669	0.438	0.772	0.394	0.393*	0.417	0.463	0.479	0.312	*0.152*	0.585	*0.616*	0.341*	3.29
h-100	0.356	0.440	0.281	0.539	0.327	0.204	0.378	0.317	0.423	0.252	0.040	0.476	0.289	0.193	4.79
h-200	*0.485*	0.539	*0.347*	0.653	*0.444*	0.287	*0.449*	0.401	0.485*	0.381*	*0.072*	0.658*	0.413	0.284	*2.43*
h-300	0.462	*0.572*	0.362*	0.678	0.400	*0.287*	0.404	0.399	*0.440*	0.301	0.080*	0.584	0.442	*0.285*	2.57
h-400	0.476	0.552	0.338	0.697*	0.398	*0.287*	0.390	*0.411*	0.429	0.283	0.080*	0.577	*0.456*	0.278	2.86
h-500	0.525*	0.591*	0.328	*0.694*	0.445*	0.311*	0.467*	0.434*	0.485*	*0.348*	0.065	*0.642*	0.473*	0.318	1.57*

Table 4.14. MRR-10 values of word2vec vectors for noun inflections in which inflection categories are denoted by the example of Turkish nouns “konu” and “gün”.

Corpus	konular	konumuz	konum	konunuz	konun	konuları	konusu	konudan	konuyu	konuya	günce	konunun	konuyla	konuda	Average
w-100	0.375	0.043	0.086	0.036*	0.436	0.313	0.484	0.250	0.463	0.384	*0.000*	0.521	0.169	0.182	4.07
w-200	0.457*	0.052	0.116	0.036*	*0.454*	0.338*	0.573*	0.260	*0.532*	0.396	*0.000*	*0.576*	0.194*	0.191	2.07
w-300	*0.455*	*0.055*	0.131*	0.036*	0.455*	0.314	0.555	0.267	0.531	0.411*	*0.000*	0.577*	*0.181*	0.208*	1.86*
w-400	0.448	0.062*	*0.121*	0.036*	0.449	*0.316*	*0.566*	0.272*	0.537*	0.384	0.009*	0.561	*0.181*	*0.193*	*2.00*
w-500	0.418	0.052	0.108	0.036*	0.414	0.277	0.526	*0.271*	0.511	*0.404*	0.009*	0.539	0.170	0.172	3.36
b-100	0.458	0.341	0.265	0.259*	0.372	0.322	0.510	0.337	0.540	0.437	*0.029*	0.650	0.342	0.227	4.29
b-200	0.520	0.371	0.315*	*0.257*	0.401	0.383*	0.559	*0.357*	0.591	0.452	*0.029*	0.664	0.423	*0.246*	2.71
b-300	*0.549*	0.379*	0.286	0.239	*0.412*	*0.381*	0.586	0.363*	0.614	0.465*	0.033*	0.676	0.456*	0.237	*2.14*
b-400	0.553*	0.363	*0.295*	0.246	0.418*	0.364	*0.622*	0.344	*0.637*	*0.462*	*0.029*	0.682*	*0.436*	0.255*	2.07*
b-500	0.541	*0.373*	0.285	0.235	*0.412*	0.347	0.623*	0.327	0.647*	0.451	0.033*	*0.678*	0.416	*0.246*	2.86
h-100	0.370	0.352	0.182	0.289	0.307	0.234	0.419	0.304	0.448	0.292	*0.018*	0.507	0.253	0.184	4.57
h-200	0.441	0.357	*0.214*	*0.292*	0.335	0.263	*0.435*	0.389*	0.471	0.359	*0.018*	0.559	0.323	0.213	3.21
h-300	0.446	0.382*	0.224*	0.300*	*0.377*	0.276	0.434	*0.376*	0.485	0.397	0.025*	*0.605*	0.369	0.244	*2.21*
h-400	*0.480*	*0.370*	0.212	0.267	0.366	*0.304*	0.429	0.371	*0.506*	*0.410*	*0.018*	0.599	*0.371*	*0.255*	2.57
h-500	0.497*	0.362	0.202	0.266	0.379*	0.309*	0.472*	0.371	0.542*	0.432*	*0.018*	0.622*	0.379*	0.262*	1.86*

Table 4.15. MRR-10 values of fastText vectors for noun inflections in which inflection categories are denoted by the example of Turkish nouns “konu” and “gün”.

Corpus	konular	konumuz	konum	konunuz	konun	konuları	konusu	konudan	konuyu	konuya	günce	konunun	konuyla	konuda	Average
w-100	0.603	0.468	0.291	0.607	0.597	0.656	0.630	0.719	0.590	0.597	0.295	0.769	0.587	0.543	4.71
w-200	0.730	*0.554*	0.431	*0.643*	0.670	0.773	0.712	0.827	0.679	0.685	0.505	0.860	0.757	0.674	3.64
w-300	0.780	0.575*	0.475	0.679*	0.704*	0.805	0.749	0.845	0.712	0.691	*0.524*	0.882*	0.816*	0.736	2.21
w-400	*0.795*	0.541	*0.483*	0.571	0.704*	*0.806*	0.765*	*0.852*	*0.737*	0.717*	0.514	*0.875*	*0.807*	0.768*	*2.00*
w-500	0.798*	0.575*	0.537*	0.500	*0.696*	0.828*	*0.760*	0.863*	0.744*	*0.714*	0.610*	0.872	0.805	*0.764*	1.86*
b-100	0.512	0.649	0.520	0.772	0.409	0.441	0.522	0.491	0.559	*0.439*	0.176	0.677	0.519	0.344	4.57
b-200	0.607	0.726	0.558	0.828	*0.467*	0.459	0.548*	0.566	0.591*	0.452*	*0.219*	0.714*	0.647	0.386	2.71
b-300	0.616	*0.754*	*0.577*	*0.831*	0.465	0.511	*0.531*	0.571	*0.579*	0.432	0.186	0.714*	0.700	0.425	*2.57*
b-400	0.630*	0.770*	0.585*	0.844*	0.460	*0.515*	0.525	0.602*	0.577	0.425	0.195	*0.708*	0.722*	0.458*	2.00*
b-500	*0.616*	0.749	0.551	*0.831*	0.479*	0.525*	0.517	*0.576*	0.574	0.412	0.233*	0.686	*0.712*	*0.450*	2.64
h-100	0.424	0.521	0.353	0.633	0.382	0.276	0.450	0.382	0.497	0.333	0.054	0.558	0.353	0.258	5.00
h-200	0.554	0.614	*0.442*	0.723	*0.501*	*0.378*	*0.533*	0.493	*0.570*	0.469*	0.105	*0.710*	0.489	0.364	*2.71*
h-300	0.542	*0.639*	0.445*	0.736	0.454	0.356	0.465	0.490	0.531	0.381	0.130*	0.643	0.515	*0.376*	2.86
h-400	*0.559*	0.622	0.427	0.760*	0.456	0.366	0.463	*0.504*	0.517	0.369	0.098	0.638	*0.538*	0.375	3.07
h-500	0.609*	0.658*	0.431	*0.759*	0.514*	0.395*	0.561*	0.556*	0.595*	*0.441*	*0.123*	0.720*	0.567*	0.408*	1.36*

Table 4.16. MRR values of context independent ELMo vectors for noun inflections in which inflection categories are denoted by the example of Turkish nouns “konu” and “gün”.

Corpus	konular	konumuz	konum	konunuz	konun	konuları	konusu	konudan	konuyu	konuya	günce	konunun	konuyla	konuda	Average
w-mrr1	0.675*	0.283*	0.177*	0.107	0.187*	0.146*	0.150	0.599*	0.157*	0.363*	0.086*	0.237*	0.499*	0.427*	1.14*
b-mrr1	0.486	0.179	0.123	0.174*	*0.129*	*0.105*	*0.088*	*0.353*	0.103	0.239	0.086*	*0.150*	0.404	0.314	*2.36*
h-mrr1	*0.490*	*0.179*	*0.136*	*0.161*	0.117	0.104	0.086	0.342	*0.104*	*0.246*	*0.069*	0.149	*0.406*	*0.315*	*2.36*
w-mrr5	0.803*	0.326*	0.273*	0.214	0.328*	0.214*	0.266*	0.794*	0.283*	0.541*	0.191*	0.358*	0.681*	0.727*	1.14*
b-mrr5	*0.737*	*0.305*	0.240	0.268*	*0.291*	*0.174*	*0.198*	*0.730*	*0.217*	*0.452*	*0.143*	*0.318*	*0.631*	*0.643*	*2.00*
h-mrr5	0.735	*0.305*	*0.243*	*0.262*	0.269	0.168	0.193	0.715	0.213	0.443	0.134	0.309	0.621	0.636	*2.79*
w-mrr10	0.840*	0.339*	0.333*	0.214	0.374*	0.235*	0.317*	0.837*	0.333*	0.605*	0.238*	0.378*	0.748*	0.781*	1.14*
b-mrr10	*0.795*	*0.337*	*0.307*	*0.296*	*0.360*	*0.197*	*0.253*	*0.784*	*0.270*	*0.530*	*0.191*	*0.350*	*0.690*	*0.730*	*2.00*
h-mrr10	0.787	0.329	0.296	0.298*	0.346	0.194	0.238	0.776	0.257	0.520	0.181	0.344	0.673	0.720	*2.86*

4.1.3. Semantic and Syntactic Analogy Task

In Table 4.17, word vectors are compared with each other on semantic and syntactic analogy test sets created in [19]. sen-200 with word vectors in dimension 200 denotes the reported result in their work. The authors have also shared their word vectors in dimension 100 which is named as sen-100 which we also used. Facebook [15] and Huawei [39] models are pretrained and publicly available which we also tested.

In Table 4.18, word2vec vectors trained on WikipediaCorpus, BounWebCorpus, and HuaweiCorpus are used for testing on semantic and syntactic analogy task. Wiki vectors in 200 and 300 dimensions are good at the relationship category. For Boun and Huawei vectors, 300 dimensional vectors are seen useful. Yet, 400 dimensional vectors are relatively worse than 300 and 500 dimensional vectors.

In Table 4.19, fastText vectors trained on WikipediaCorpus, BounWebCorpus, and HuaweiCorpus are used for testing on semantic and syntactic analogy tasks. It is observed that fastText vectors are behind the word2vec vectors. FastText vectors show better performance in the 200 dimensional space. For the category "capital", the performance of wiki vectors dramatically decreases despite increasing dimensionality. A similar behaviour is seen with Wiki and Boun vectors in the category "opposite".

In Table 4.20, context-independent ELMo vectors are obtained by using vocabularies of WikipediaCorpus, BounWebCorpus, and HuaweiCorpus. They are used for testing on the semantic and syntactic analogy tasks. The best performing model belongs to the one using the vocabulary of WikipediaCorpus while the worst one is trained on the HuaweiCorpus.

Table 4.17. Semantic and syntactic accuracy results on pre-trained vectors obtained from references.

Model	Accuracy(%)										
	relationship	capital	synonym	district	currency	opposite	plural	past tense	negative	present tense	Average
huawei [39]	8.33	37.64	22.12	34.01	7.69	15.78	39.67	35.56	45.37	49.74	33.18
facebook	2.27	46.84	5.67	13.45	3.21	12.74	27.76	24.05	24.87	35.02	21.54
sen-200 [19]	-	-	-	-	-	-	-	-	-	-	34.74
sen-100 [19]	9.85	43.47	18.35	27.40	12.82	16.07	36.46	40.85	28.44	46.86	31.29

Table 4.18. Semantic and syntactic accuracy results on word2vec vectors in different dimensions on the three corpora.

Model	Accuracy(%)										
	relationship	capital	synonym	district	currency	opposite	plural	past tense	negative	present tense	Average
w-100	9.09	36.3	23.61	24.03	23.88	22.54	22.16	23.21	22.95	23.38	23.11
w-200	13.64*	37.98*	25.28*	*27.63*	27.42*	26.39*	*25.51*	26.25*	*25.95*	*26.35*	*26.24*
w-300	13.64*	*36.98*	*24.92*	27.12	*26.92*	*25.97*	25.11	*25.46*	25.2	25.47	25.67
w-400	*12.12*	32.72	22.65	23.61	23.45	22.9	22.56	22.81	22.59	22.94	22.83
w-500	6.82	36.73	21.04	45.76*	2.56	17.82	34.47*	24.10	41.53*	34.23*	32.50*
b-100	9.85*	34.56	23.94	25.77	25.54	23.38	25.08	25.0	25.28	25.6	24.4
b-200	9.85*	42.17*	*30.04*	36.22	35.84	32.64	33.13	*32.86*	33.03	33.31	31.90
b-300	8.33	*41.68*	30.86*	*38.44*	38.01*	34.55*	34.82*	33.99*	*34.15*	34.5*	32.93*
b-400	*9.09*	39.07	29.61	37.24	*36.83*	*33.76*	33.95	32.83	33.02	33.18	31.85
b-500	6.82	36.73	21.04	45.76*	2.56	17.82	*34.47*	24.10	41.53*	*34.23*	*32.50*
h-100	*2.27*	38.17	21.75	27.47	27.25	24.07	22.3	21.33	21.46	21.58	22.76
h-200	3.03*	49.13	*28.86*	43.83	*43.44*	*37.75*	*33.37*	*31.39*	31.5	31.57	33.38
h-300	*2.27*	52.77	32.05*	48.91	48.43*	42.29*	36.89*	34.58*	34.76	34.79	36.77*
h-400	1.52	*55.79*	14.06	*66.41*	7.69	13.61	19.48	19.58	*41.93*	*36.79*	35.15
h-500	1.52	58.08*	14.85	69.24*	6.41	13.10	19.54	18.90	45.37*	37.82*	*36.22*

Table 4.19. Semantic and syntactic accuracy results on fastText vectors in different dimensions on the three corpora.

Model	Accuracy(%)										
	relationship	capital	synonym	district	currency	opposite	plural	past tense	negative	present tense	Average
w-100	0.76	33.67	5.49	14.74	1.28	10.30	34.04	31.06	47.88	39.88	*21.91*
w-200	0.76	29.49	4.88	14.21	1.92	12.30	44.24	30.68	56.35	36.91	23.17*
w-300	0.00	18.45	3.74	11.18	1.28	11.10	49.25	29.25	55.95	30.84	21.10
w-400	0.00	9.46	3.21	4.96	0.64	9.11	49.92	21.62	57.01	21.46	17.73
w-500	0.00	5.05	2.54	2.27	0.00	6.75	49.69	17.30	51.32	15.79	15.07
b-100	5.30	33.23	7.98	23.09	1.28	11.47	31.55	16.61	35.19	30.06	19.57
b-200	5.30	36.80	9.38	23.18	1.28	12.99	37.74	19.24	49.74	36.15	23.18*
b-300	4.55	30.98	8.62	18.93	1.28	11.65	37.56	17.26	51.06	34.74	*21.66*
b-400	4.55	21.25	7.60	12.54	1.28	9.47	35.47	15.37	50.40	32.50	19.04
b-500	5.30	15.56	7.45	8.44	1.92	7.84	30.75	12.77	47.62	26.86	16.45
h-100	1.52	39.06	5.46	30.31	7.05	7.37	17.93	10.76	25.79	23.53	16.87
h-200	3.03	53.06	8.83	50.17	5.77	11.07	31.90	19.12	34.92	38.14	25.60*
h-300	2.27	50.74	7.83	42.39	3.21	10.45	20.77	10.28	39.95	27.69	21.55
h-400	2.27	46.16	8.04	39.02	5.13	10.01	20.97	8.95	43.12	27.12	21.07
h-500	7.58	40.37	7.86	40.46	6.41	11.57	35.84	16.47	45.24	34.55	*24.63*

Table 4.20. Semantic and syntactic accuracy results on context-independent ELMo vectors on the three corpora.

Model	Accuracy(%)										
	relationship	capital	synonym	district	currency	opposite	plural	past tense	negative	present tense	Average
w	3.03	1.25	2.22	2.58	4.49	1.16	55.57	16.57	14.81	9.04	14.47*
b	2.27	0.98	1.34	1.82	1.28	0.73	37.93	13.05	9.26	7.44	*10.15*
h	1.52	0.91	1.08	1.22	0.64	0.44	37.33	11.89	11.38	7.18	9.70

4.2. Classification Tasks

First, word embedding models are trained on the Turkish corpora as explained in detail in Section 3, to learn the word embeddings. Once word vectors with different dimensions are learnt, the vector of a document is determined by averaging the word vectors. We also use the bag-of-words model as another feature extraction method. After the features are extracted for each news article, a linear classifier is trained in order to evaluate them. In the tables shown in this section, the best test result is shown with an asterisk.

4.2.1. Hurriyet News

Determining the class labels of the News articles does not require all the words. In our study, the first n words are extracted from each article, since the length of each article in Hurriyet News dataset varies. In the experiment, the bag-of-words model is utilized in which the bag size is set to 2000.

Figure 4.2 shows how the number of words in articles affect the accuracy results on the classification task. After conducting the experiment, n is set to 100. Furthermore, the best result is obtained by the fastText model, *h-500*.

In Table 4.21, BOW features are evaluated on the classification of Hurriyet News dataset. It is observed that increasing the length of BOW features results in better performances.

In Table 4.22, ELMo features are used where layer-0 is context-independent features, layer-1 and layer-2 are the LSTM layers above it. It is seen that the last layer features are more informative and successful than the others.

In Table 4.23, word2vec vectors are used for representing documents. Increasing the vector dimensionality leads to better accuracy results.

In Table 4.24, fastText vectors are used for representing documents. The best result is obtained by a model trained on huawei corpus with 500 dimensions.

In Tables 4.25, 4.26, and 4.27, clustering based approach on the classification task are given. The best result is obtained by h-200 where the cluster size is 1000. Increasing the cluster size generally results in better accuracy.

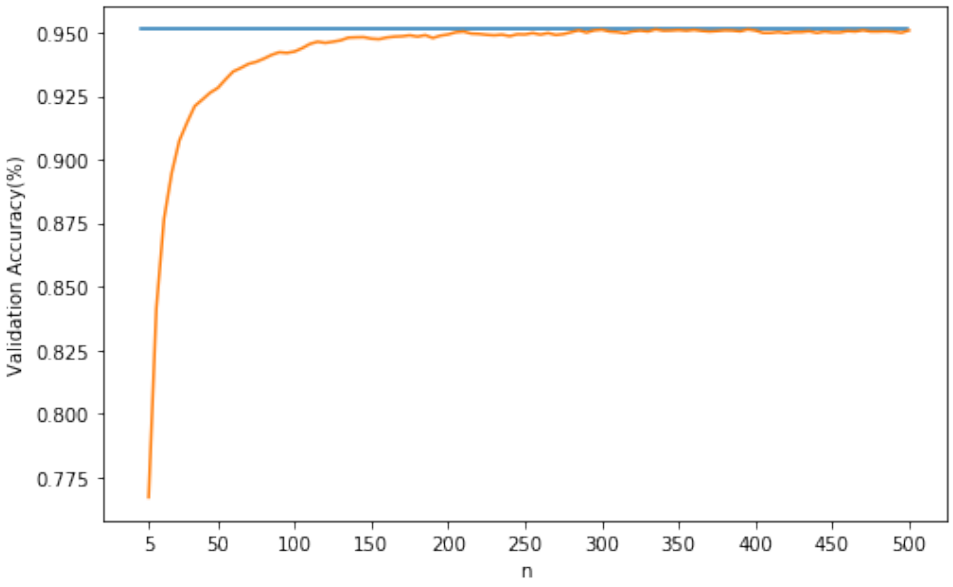

Figure 4.2. Effects of the first n words are observed by computing accuracy results on Hurriyet News dataset.

In summary, we have obtained the best test results with 500 dimensional vectors on Huawei data. Increasing the dimensionality of vectors results in better accuracy. BOW models in higher dimensionality perform better than the clustering models.

Table 4.21. Accuracy results of BOW models on Hurriyet News dataset.

Model	Feature Length	Train	Validation	Test
bow	100	0.8432	0.8466	0.8386
bow	500	0.9295	0.9236	0.9185
bow	2000	0.9639	0.9513	0.9517
bow	10000	0.9812	0.9605	0.9605
bow	20000	0.9872	0.9630	0.9612*

Table 4.22. Accuracy results of features extracted from the last three ELMo layers on Hurriyet News dataset.

Model	Feature Length	Train	Validation	Test
layer-0	256	0.8770	0.8681	0.8584
layer-1	256	0.9165	0.9003	0.8967
layer-2	256	0.9209	0.9100	0.9072

Table 4.23. Accuracy results of word2vec models on Hurriyet News dataset.

Model	Feature Length	Train	Validation	Test
w-sum	100	0.9440	0.9460	0.9420
w-sum	200	0.9498	0.9511	0.9477
w-sum	300	0.9498	0.9523	0.9486
w-sum	400	0.9533	0.9539	0.9520*
w-sum	500	0.9509	0.9513	*0.9498*
b-sum	100	0.9479	0.9496	0.9465
b-sum	200	0.9542	0.9564	0.9533
b-sum	300	0.9588	0.9588	0.9549
b-sum	400	0.9583	0.9597	0.9559*
b-sum	500	0.9577	0.9593	*0.9553*
h-sum	100	0.9493	0.9515	0.9476
h-sum	200	0.9573	0.9581	0.9551
h-sum	300	0.9594	0.9609	0.9575
h-sum	400	0.9605	0.9614	*0.9585*
h-sum	500	0.9619	0.9621	0.9595*

Table 4.24. Accuracy results of fastText models on Hurriyet News dataset.

Model	Feature Length	Train	Validation	Test
w	100	0.9470	0.9435	0.9471
w	200	0.9537	0.9483	0.9527
w	300	0.9554	0.9502	0.9530
w	400	0.9563	0.9509	*0.9540*
w	500	0.9590	0.9523	0.9561*
b	100	0.9535	0.9494	0.9527
b	200	0.9588	0.9552	0.9588
b	300	0.9618	0.9561	0.9609*
b	400	0.9599	0.9560	0.9602
b	500	0.9634	0.9573	*0.9606*
h	100	0.9547	0.9494	0.9559
h	200	0.9611	0.9558	0.9608
h	300	0.9622	0.9579	0.9622
h	400	0.9631	0.9579	*0.9637*
h	500	0.9649	0.9598	0.9642*

Table 4.25. Classification results of the clustering based feature extraction approach using Wiki vectors on Hurriyet News.

model	vector size	cluster size	training accuracy	validation accuracy	test accuracy
w	100	500	0.9532	0.9436	0.9494
w	100	1000	0.9565	0.9468	0.9526
w	100	2000	0.9616	0.9499	*0.9533*
w	200	500	0.9515	0.9422	0.9485
w	200	1000	0.9483	0.9416	0.9445
w	200	2000	0.9594	0.9479	0.9499
w	300	500	0.9369	0.9309	0.9319
w	300	1000	0.9516	0.9452	0.9476
w	300	2000	0.9613	0.9482	0.9543*
w	400	500	0.9528	0.9455	0.9483
w	400	1000	0.9483	0.9387	0.9423
w	400	2000	0.9572	0.9487	0.9523
w	500	500	0.9488	0.9425	0.9468
w	500	1000	0.9549	0.9455	0.9496
w	500	2000	0.9582	0.9469	0.9511

Table 4.26. Classification results of the clustering based feature extraction approach using Boun vectors on Hurriyet News.

model	vector size	cluster size	training accuracy	validation accuracy	test accuracy
b	100	500	0.9575	0.9488	0.9520
b	100	1000	0.9563	0.9452	0.9479
b	100	2000	0.9612	0.9500	0.9549
b	200	500	0.9357	0.9288	0.9325
b	200	1000	0.9596	0.9502	0.9551
b	200	2000	0.9630	0.9530	0.9565*
b	300	500	0.9547	0.9479	0.9510
b	300	1000	0.9593	0.9515	*0.9554*
b	300	2000	0.9638	0.9510	0.9547
b	400	500	0.9210	0.9163	0.9198
b	400	1000	0.9446	0.9366	0.9419
b	400	2000	0.9596	0.9499	0.9543
b	500	500	0.9520	0.9459	0.9493
b	500	1000	0.9557	0.9485	0.9522
b	500	2000	0.9599	0.9497	0.9549

Table 4.27. Classification results of the clustering based feature extraction approach using Huawei vectors on Hurriyet News.

model	vector size	cluster size	training accuracy	validation accuracy	test accuracy
h	100	500	0.9563	0.9491	0.9529
h	100	1000	0.9623	0.9517	0.9575
h	100	2000	0.9660	0.9533	*0.9585*
h	200	500	0.9568	0.9517	0.9538
h	200	1000	0.9629	0.9535	0.9587*
h	200	2000	0.9615	0.9516	0.9563
h	300	500	0.9455	0.9427	0.9451
h	300	1000	0.9607	0.9525	0.9550
h	300	2000	0.9640	0.9537	0.9577
h	400	500	0.9570	0.9497	0.9521
h	400	1000	0.9593	0.9517	0.9551
h	400	2000	0.9549	0.9463	0.9463
h	500	500	0.9542	0.9504	0.9531
h	500	1000	0.9590	0.9501	0.9543
h	500	2000	0.9508	0.9342	0.9396

4.2.2. Kemik News

On the Kemik News data [40], stop-words in articles are removed, then features of each article is determined. For the BoW model, the authors created a word list where words are sorted by their tf-idf values, they used the first 500 and 2000 words. The best result is denoted with an asterisk and in italic results denote the best results for each vector model. In Table 4.28, results obtained from the reference is represented.

In Table 4.29, ELMo features are used where layer-0 is context-independent features, layer-1 and layer-2 are above it. It is such that the last layer features are more informative and successful than the others.

In Table 4.30, BOW features are evaluated on the classification of Kemik News dataset. It is observed that increasing the length of BOW features results in better performances.

In Table 4.31, word2vec vectors are used for representing documents. Increasing the vector dimensionality causes better accuracy results with Boun and Huawei vectors.

In Table 4.32, fastText vectors are used for representing documents. The best result is obtained by a model trained on the Huawei corpus with 500 dimensions. Similar to word2vec models, it is better to increase the vector dimensions for obtaining better results.

In Tables 4.33, 4.34, and 4.35, the clustering based approach is tested on the classification task. The best results are obtained by h-100 and h-400 where the cluster size is 2000. Increasing the cluster size generally results in better accuracy.

In summary, our BOW models in Table 4.30 are better than the reference results in Table 4.28. The best result is obtained with 400 dimensional word2vec vectors on Huawei data. The clustering approach generally shows better performance when the cluster size is increased.

Table 4.28. Accuracy results of reference models on Kemik News dataset.

Model	Feature Length	Train	Validation	Test
SG_HS [40]	400	-	-	0.91
SG_NS [40]	400	-	-	0.91
CBoW_HS [40]	400	-	-	0.90
CBoW_NS [40]	400	-	-	0.91
BoW(tf-idf) [40]	500	-	-	0.87
BoW(tf-idf) [40]	2000	-	-	0.89

Table 4.29. Accuracy results of features extracted from the last three ELMo layers on Kemik News dataset.

Model	Feature Length	Train	Validation	Test
layer-0	256	0.8737	0.8585	0.8619
layer-1	256	0.9092	0.8958	0.8908
layer-2	256	0.9178	0.9043	0.8964

Table 4.30. Accuracy results of BOW models on Kemik News dataset.

Model	Feature Length	Train	Validation	Test
bow	100	0.7318	0.7249	0.7179
bow	500	0.9068	0.8676	0.8571
bow	2000	0.9747	0.9214	0.9156
bow	10000	0.9959	0.9385	*0.9371*
bow	20000	0.9990	0.9391	0.9374*

Table 4.31. Accuracy results of word2vec models on Kemik News dataset.

Model	Feature Length	Train	Validation	Test
w	100	0.9300	0.9314	0.9284
w	200	0.9353	0.9332	0.9308
w	300	0.9410	0.9351	0.9329*
w	400	0.9372	0.9346	0.9312
w	500	0.9402	0.9361	*0.9314*
b	100	0.9333	0.9297	0.9293
b	200	0.9365	0.9349	0.9359
b	300	0.9415	0.9397	0.9369
b	400	0.9515	0.9442	*0.9395*
b	500	0.9507	0.9412	0.9397*
h	100	0.9372	0.9363	0.9320
h	200	0.9411	0.9404	0.9344
h	300	0.9517	0.9463	0.9410
h	400	0.9538	0.9464	0.9444*
h	500	0.9522	0.9474	*0.9435*

Table 4.32. Accuracy results of fastText models on Kemik News dataset.

Model	Feature Length	Train	Validation	Test
w	100	0.9277	0.9304	0.9252
w	200	0.9343	0.9348	0.9303
w	300	0.9368	0.9351	0.9288
w	400	0.9397	0.9359	*0.9312*
w	500	0.9409	0.9363	0.9346*
b	100	0.9355	0.9332	0.9308
b	200	0.9440	0.9391	0.9387
b	300	0.9431	0.9381	0.9340
b	400	0.9495	0.9398	*0.9376*
b	500	0.9566	0.9453	0.9395*
h	100	0.9341	0.9340	0.9274
h	200	0.9485	0.9440	0.9371
h	300	0.9463	0.9427	0.9389
h	400	0.9507	0.9461	*0.9421*
h	500	0.9538	0.9476	0.9431*

Table 4.33. Classification results of the clustering based feature extraction approach using Wiki vectors on Kemik News.

model	vector size	cluster size	training accuracy	validation accuracy	test accuracy
w	100	500	0.9255	0.9191	0.9118
w	100	1000	0.9326	0.9225	0.9156
w	100	2000	0.9476	0.9287	0.9208
w	200	500	0.9129	0.9106	0.9056
w	200	1000	0.9460	0.9283	*0.9240*
w	200	2000	0.9373	0.9204	0.9182
w	300	500	0.9279	0.9197	0.9169
w	300	1000	0.9213	0.9133	0.9093
w	300	2000	0.9504	0.9299	0.9195
w	400	500	0.9347	0.9236	0.9190
w	400	1000	0.9282	0.9161	0.9086
w	400	2000	0.9453	0.9244	0.9190
w	500	500	0.9004	0.9016	0.8980
w	500	1000	0.9192	0.9116	0.9012
w	500	2000	0.9599	0.9372	0.9259*

Table 4.34. Classification results of the clustering based feature extraction approach using Boun vectors on Kemik News.

model	vector size	cluster size	training accuracy	validation accuracy	test accuracy
b	100	500	0.9186	0.9093	0.9048
b	100	1000	0.9428	0.9310	*0.9254*
b	100	2000	0.9442	0.9291	0.9233
b	200	500	0.8920	0.8927	0.8848
b	200	1000	0.9266	0.9144	0.9084
b	200	2000	0.9419	0.9293	0.9176
b	300	500	0.9312	0.9248	0.9154
b	300	1000	0.9089	0.9035	0.8999
b	300	2000	0.9279	0.9227	0.9144
b	400	500	0.8775	0.8727	0.8713
b	400	1000	0.9286	0.9223	0.9169
b	400	2000	0.9442	0.9308	0.9256*
b	500	500	0.8704	0.8708	0.8605
b	500	1000	0.8974	0.8872	0.8818
b	500	2000	0.9378	0.9302	0.9161

Table 4.35. Classification results of the clustering based feature extraction approach using Huawei vectors on Kemik News.

model	vector size	cluster size	training accuracy	validation accuracy	test accuracy
h	100	500	0.9352	0.9287	*0.9237*
h	100	1000	0.9285	0.9212	0.9142
h	100	2000	0.9480	0.9317	0.9269*
h	200	500	0.9347	0.9255	0.9175
h	200	1000	0.9396	0.9270	0.9231
h	200	2000	0.9394	0.9195	0.9154
h	300	500	0.9178	0.9136	0.9065
h	300	1000	0.9228	0.9200	0.9139
h	300	2000	0.9398	0.9340	0.9220
h	400	500	0.9316	0.9257	0.9227
h	400	1000	0.9332	0.9263	0.9205
h	400	2000	0.9522	0.9349	0.9269*
h	500	500	0.9133	0.9144	0.9014
h	500	1000	0.9240	0.9238	0.9146
h	500	2000	0.9395	0.9278	*0.9237*

5. CONCLUSIONS

5.1. Summary of findings

In this study, various word embedding models on Turkish have been tested by conducting several experiments on noun and verb inflection tasks, syntactic and semantic analogy tests, and classification of news documents. Word2vec, fastText, and ELMO are used in experiments. The most important points obtrained from this study are as follows:

- Vectors in different dimensions can lead to different performances on the analogy tasks. Yet, words having a specific inflection are learned better in some vector dimensionalities.
- BOW models show better performance on classification than the most of the word embedding models.
- It is observed that a fastText model shows better performance for learning syntactic features, namely, noun and verb inflections.
- Semantic features are learned better by the word2vec model.
- For out-of-vocaulary (OOV) words, fastText and ELMo are useful due to their capability of handling subword information.
- ELMo features are useful for tasks that require the contextual information
- Training an ELMo model takes longer than other word embedding models.

5.2. Future work

- As a future work, model ensembling methods can be utilized in order to obtain better word representations since we have shown that each vector model can be superior to other ones. Therefore, combining different vector models may boost the overall performance.
- In addition to methods mentioned in the study, BERT and GloVe can be used for experiments.

- Furthermore, different analogy tasks can be created in order to evaluate word embedding models on Turkish.
- Different tests can be used for evaluating word embedding models such as question-answering, machine translation, and so on.

REFERENCES

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

2. Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, "Deep contextualized word representations", *arXiv preprint arXiv:1802.05365*, 2018.

3. Pennington, J., R. Socher and C. Manning, "Glove: Global vectors for word representation", *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

4. *Google Code Archive - Long-term storage for Google Code Project Hosting*, 2018, `https://code.google.com/archive/p/word2vec`, accessed at December 2018.

5. Alpaydın, E., *Introduction to machine learning*, MIT press, 2008.

6. McCulloch, W. S. and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, Vol. 5, No. 4, pp. 115–133, 1943.

7. Rumelhart, D. E., G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors", *nature*, Vol. 323, No. 6088, p. 533, 1986.

8. Hochreiter, S. and J. Schmidhuber, "Long short-term memory", *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

9. Pascanu, R., T. Mikolov and Y. Bengio, "On the difficulty of training recurrent neural networks", *International Conference on Machine Learning*, pp. 1310–1318, 2013.

10. Cho, K., B. Van Merriënboer, D. Bahdanau and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches", *arXiv preprint arXiv:1409.1259*, 2014.

11. Chung, J., C. Gulcehre, K. Cho and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling", *arXiv preprint arXiv:1412.3555*, 2014.

12. Sutskever, I., O. Vinyals and Q. V. Le, "Sequence to sequence learning with neural networks", *Advances in neural information processing systems*, pp. 3104–3112, 2014.

13. Mikolov, T., M. Karafiát, L. Burget, J. Černockỳ and S. Khudanpur, "Recurrent neural network based language model", *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

14. Graves, A., A.-r. Mohamed and G. Hinton, "Speech recognition with deep recurrent neural networks", *IEEE international conference on Acoustics, speech and signal processing (icassp), 2013*, pp. 6645–6649, IEEE, 2013.

15. Bojanowski, P., E. Grave, A. Joulin and T. Mikolov, "Enriching Word Vectors with Subword Information", *arXiv preprint arXiv:1607.04606*, 2016.

16. Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality", *Advances in neural information processing systems*, pp. 3111–3119, 2013.

17. Hartigan, J. A. and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm", *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 28, No. 1, pp. 100–108, 1979.

18. *Data repository of "Linguistic Features in Turkish Word Representations"*, 2018, `https://github.com/onurgu/`, accessed at December 2018.

19. Sen, M. U. and H. Erdogan, "Learning word representations for Turkish", *Signal Processing and Communications Applications Conference (SIU), 2014 22nd*, pp. 1742–1745, IEEE, 2014.

20. Ayata, D., M. Saraçlar and A. Özgür, "Turkish tweet sentiment analysis with word embedding and machine learning", *Signal Processing and Communications Applications Conference (SIU), 2017 25th*, pp. 1–4, IEEE, 2017.

21. Cortes, C. and V. Vapnik, "Support-vector networks", *Machine learning*, Vol. 20, No. 3, pp. 273–297, 1995.

22. Liaw, A., M. Wiener *et al.*, "Classification and regression by randomForest", *R news*, Vol. 2, No. 3, pp. 18–22, 2002.

23. Üstün, A., M. Kurfalı and B. Can, "Characters or Morphemes: How to Represent Words?", *Proceedings of The Third Workshop on Representation Learning for NLP*, pp. 144–153, 2018.

24. Gungor, O., E. Yildiz, S. Uskudarli and T. Gungor, "Morphological Embeddings for Named Entity Recognition in Morphologically Rich Languages", *arXiv preprint arXiv:1706.00506*, 2017.

25. Ercan, G. and O. T. Yıldız, "AnlamVer: Semantic Model Evaluation Dataset for Turkish-Word Similarity and Relatedness", *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 3819–3836, 2018.

26. Ertopcu, B., A. B. Kanguroglu, O. Topsakal *et al.*, "A New Approach for Named Entity Recognition", *International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey*, pp. 474–479, 2017.

27. Topsakal, O., O. Açıkgöz, A. T. Gürkan *et al.*, "Shallow parsing in Turkish", *2017 International Conference on Computer Science and Engineering (UBMK)*, pp. 480–485, Oct 2017.

28. Acikgoz, O., A. Tunca Gurkan, B. Ertopcu, O. Topsakal, B. Ozenc, A. B. Kanburoglu, I. Cam, B. Avar, G. Ercan and O. Yildiz, "All-words word sense disambiguation for Turkish", *2017 International Conference on Computer Science and Engineering (UBMK)*, pp. 490–495, 10 2017.

29. *Wikimedia Downloads*, 2018, `https://dumps.wikimedia.org/`, accessed at December 2018.

30. *WikiExtractor*, 2018, `https://github.com/attardi/wikiextractor`, accessed at December 2018.

31. Sak, H., T. Güngör and M. Saraçlar, "Turkish language resources: Morphological parser, morphological disambiguator and web corpus", *Advances in natural language processing*, pp. 417–427, 2008.

32. Yildiz, E., C. Tirkaz, H. B. Sahin, M. T. Eren and O. Sonmez, "A morphology-aware network for morphological disambiguation", *arXiv preprint arXiv:1702.03654*, 2017.

33. *Hürriyet Public API*, 2018, `https://developers.hurriyet.com.tr`, accessed at December 2018.

34. Chollet, F. *et al.*, *Keras*, 2015, `https://keras.io`, accessed at December 2018.

35. Abadi, M., A. Agarwal, P. Barham *et al.*, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015, `https://www.tensorflow.org/`, accessed at December 2018.

36. Kingma, D. P. and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

37. Duchi, J., E. Hazan and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", *Journal of Machine Learning Research*, Vol. 12,

No. Jul, pp. 2121–2159, 2011.

38. Tieleman, T. and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude", *COURSERA: Neural networks for machine learning*, Vol. 4, No. 2, pp. 26–31, 2012.

39. Güngör, O. and E. Yıldız, "Linguistic features in Turkish word representations", *25th Signal Processing and Communications Applications Conference (SIU), 2017*, pp. 1–4, IEEE, 2017.

40. Şahin, G., "Turkish document classification based on Word2Vec and SVM classifier", *Signal Processing and Communications Applications Conference (SIU), 2017 25th*, pp. 1–4, IEEE, 2017.