CmpE 492

Building a Word Embeddings Repository for Turkish

Cahid Arda Öz (2019400132)

Karahan Sarıtaş (2018400174)

Advisor: Tunga Güngör

April 9, 2023

TABLE OF CONTENTS

1. Introduction

1.1. Broad Impact

Building a comprehensive word embedding repository for Turkish using different state-of-the-art methods can have a significant impact on various natural language processing (NLP) applications for the Turkish language. Word embeddings are a powerful tool in NLP that can represent the meaning and context of words in a language. By capturing the semantic relationships between words, word embeddings can be used in a wide range of applications such as text classification, sentiment analysis, machine translation, and question-answering systems.

By building a word embedding repository for Turkish using different methods, we can improve the quality of existing NLP applications and enable the development of new ones. This can have a direct impact on industries such as e-commerce, finance, and healthcare, where the Turkish language is commonly used. For example, accurate sentiment analysis in Turkish can help companies understand customer feedback and improve their products and services accordingly. Similarly, machine translation between Turkish and other languages can facilitate cross-cultural communication and trade.

Moreover, building a word embedding repository for Turkish can also benefit research in NLP and related fields. Availability of a high-quality word embedding repository for Turkish can enable researchers to conduct more accurate and effective studies on various topics such as language modeling, machine learning, and data mining.

Overall, building a comprehensive word embedding repository for Turkish using different methods can have a broad impact on various industries and research fields, and can contribute to developing and improving NLP applications for the Turkish language.

1.2. Ethical Considerations

AI Ethics is a code of conduct consisting of moral principles and techniques regulating AI technologies. It can be said that the ethics of AI is a brand-new topic within applied ethics with remarkable discussion, but few well-established issues and no authoritative overviews[1] . Ethics or moral philosophy is a branch of philosophy that "involves systematizing, defending, and recommending concepts of right and wrong behavior"[2] . Our intention here is not to produce *objective moral rules*, that are absolute, not contingent upon any desire or preference or policy or choice, one's own or anyone else's[3] . Undoubtedly, these questions will require further investigation, particularly in the realm of meta-ethics. Our primary objective here is to present our considerations on some of the most commonly debated ethical issues in relation to AI, such as fairness and bias.

NLP algorithms, unfortunately, can be *biased* because of the data they are trained on. For instance, language models trained on a corpus of data that contains racial bias can perpetuate those biases in their outputs. To avoid these issues, one potential solution is to use large datasets

[1]Bostrom, N., & Yudkowsky, E. (2014). The ethics of artificial intelligence. In E. N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy (Spring 2014 ed.). Stanford University. Retrieved from https://plato.stanford.edu/entries/ethics-ai/

[2]Internet Encyclopedia of Philosophy "Ethics". Archived from the original on January 18, 2018. Retrieved January 7, 2012.

[3]J. L. Mackie. Ethics: Inventing Right and Wrong. Penguin Books, London, 1977, p.33.

with the aim of reducing the bias that may be present in smaller datasets. For our project, we used a massive dataset containing an impressive number of sentences. The text corpus has been compiled from the web and contains about 500 million tokens (the largest Turkish web corpus published)[4] . Another ethical consideration can be about privacy and security of the data. NLP systems often deal with sensitive information such as personal conversations or medical records. It is essential to ensure that such information is stored and transmitted securely and that users are aware of how their data is being used. We gather all of our data from different sources that are publicly available[5] . Another concern can be the transparency of the training procedure. The training process and the sources of data used are transparent and clearly documented so that it is possible to audit and verify the results[6] .

2. Project Definition and Planning

2.1. Project Definition

In this project, we aim at building a comprehensive word embedding repository for the Turkish language. Using each of the state-of-the-art word embedding methods, embeddings of all the words in the language will be formed using a corpus. First, the three commonly-used embedding methods (Word2Vec [1, 2], Glove [3], Fasttext [4]) will be used and an embedding dictionary for each one will be formed. Then we will continue with context-dependent embedding methods such as BERT [5] and Elmo [6]. Each method will be applied with varying parameters such as different corpora and different embedding dimensions. The methods will be evaluated on analogy and similarity tasks.

The project also includes a literature survey, risk analysis, teamwork, system requirements and modeling, system architecture, and design, implementation, and testing. The success criteria of the project are based on the quality of the resulting embeddings, as well as the completion of the project within the allotted time and resources. The project has a broad impact on the field of natural language processing. It can be used in various applications such as text classification, sentiment analysis, and machine translation for the Turkish language.

2.2. Project Planning

2.2.1. Project Time and Resource Estimation

From the beginning of the semester until the midterm report, we have worked on preparing the model evaluation software package and training evaluating non-contextual models: word2vec, fasttext and GloVe. In the first weeks, we did some research on word embeddings and models we were planning to use. After that, we worked on writing the evaluation package and training word2vec and fasttext models. Last weeks before the midterm report, we worked on training GloVe. GloVe took us more than we liked because we couldn't train the model right away on the remote machine we used. We had to spend some time debugging the issue in order to finally understand that the issue was the lack of disk space.

Even though we don't have any results in our report, we have started working on contextual embedding models (BERT and Elmo). In the following weeks until the final report, we will be

[4]https://link.springer.com/article/10.1007/s10579-010-9128-6
[5]https://tulap.cmpe.boun.edu.tr/repository/xmlui/handle/20.500.12913/16
[6]https://github.com/Turkish-Word-Embeddings/Word-Embeddings-Repository-for-Turkish

working on comparing existing BERT and Elmo weights with the non-contextual models we trained. Towards the end of the semester, we are hoping to start working on putting together our results and writing a paper.

In the first weeks, we mainly used our own computers for training word embeddings. Then, we gained access to the remote machine of the faculty and we were able to train models faster. This machine allowed us to train models faster thanks to its abundence of RAM and powerful CPU. We are planning to keep using this machine for the contextual models.

2.2.2. Success Criteria

In our project, we are trying to improve the results published in papers that use the same analogy and similarity task datasets as ours. By testing several models including word2vec, fastttext, GloVe, BERT, and ELMO; we hope to achieve higher scores than the papers. Additionally, the project should be seen as a valuable contribution to the field of natural language processing (NLP) for Turkish, as it provides a comprehensive benchmark of the performance of several widely used models.

2.2.3. Risk Analysis

Our project may encounter various risks and challenges. One crucial aspect of Machine Learning applications is the quality of the data, as biased data can lead to misleading results in our comparisons. To avoid this issue, we combined the *BounWebCorpus* and *HuaweiCorpus*, resulting in a corpus of 1,384,961,747 tokens. By collecting a significant amount of text and improving data quality, we aimed to reduce potential biases.

Another challenge we may face is ensuring the accuracy of our evaluation results. To address this concern and ensure that our evaluation tasks comprehensively capture the model's performance, we compiled analogy and similarity tasks from different sources[7] [8] . These tasks encompassed 14,321 examples for verb conjugation suffixes, 14,005 examples for noun declension suffixes, 3,296 examples for semantic analogy, 500 examples for semantic similarity, and 140 examples for syntactic similarity.

Inadequate computational resources may also pose a problem for our project. To train our Word2Vec and FastText models, we upgraded to a more powerful local machine equipped with an i7-11390H Intel processor and 16 GB of memory. This resulted in a substantial reduction in algorithm execution time compared to our previous machine, which had an i5-8265U Intel processor and 8 GB of memory. For our GloVe model, we utilized a remote machine with a Slurm job scheduler. However, as all models were trained using the CPU, we were unable to utilize the GPUs provided by the remote machine.

2.2.4. Team Work

From the outset, we identified the need to efficiently train and evaluate multiple models on the same dataset. To accomplish this, we divided the task of model training between the team members, with each member responsible for training a specific set of models. This allowed us to train multiple models in parallel and significantly reduced the overall training time required for

[7] `https://github.com/onurgu/linguistic-features-in-turkish-word-representations/releases/tag/v1.0`

[8] `https://github.com/bunyamink/word-embedding-models/tree/master/datasets`

the project.

In addition to the model training, we also worked collaboratively on the development of the evaluation package. Given its high priority in the project, we allocated sufficient resources to its development and ensured that it was designed in a way that made it easy to use and allowed us to quickly iterate and produce results. This allowed us to streamline the evaluation process and provided us with a reliable way to compare the performance of the different models that we trained.

Throughout the project, we maintained regular communication and coordination to ensure that everyone was on the same page and that progress was being made according to schedule. This included regular check-ins, progress updates, and code reviews to ensure that the different parts of the project were integrating smoothly and that there were no conflicts or issues that could cause delays.

Overall, effective teamwork and collaboration between the project members were key factors in the successful completion of the project. By dividing the work, prioritizing key tasks, and maintaining regular communication and coordination, we were able to work efficiently and effectively toward our shared goals.

3. Related Work

There are various word embedding methodologies and related articles in the literature. In their first paper *Efficient estimation of word representations in vector space* [1] by Mikolov et. al., two model architectures, namely the Continuous Bag of Words and Skip-gram techniques, are introduced for computing continuous vector representations of words from very large data sets. However, the main problem with the first version of the skip-gram algorithm was the cost of computing being proportional to the number of words in the vocabulary. In the upcoming paper *Distributed Representations of Words and Phrases and their Compositionality* [2], several extensions on the previously introduced Skip-gram model are introduced to improve both the quality of the vectors and the training speed. Hierarchical softmax, negative sampling and subsampling of the frequent words are the main extensions provided in this paper. Word2Vec algorithms are implemented in various open-source projects and tutorials including Gensim[9] [7], TensorFlow[10] and other publicly available sources[11] [12] . Word2Vec is used for learning Turkish word embeddings in various articles as well [8].

GloVe: Global Vectors for Word Representation [3] is another groundbreaking paper for learning word embeddings that use two major model families in the literature: global matrix factorization and local context window methods. GloVe model is introduced[13] by Pennington et. al. and compared with other models in the literature.

In *Enriching Word Vectors with Subword Information* [4], FastText model is introduced by Bojanowski et. al. In contrast to the previous models that ignore the morphology of the

[9]Radim Rehurek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pages 45–50, Valletta, Malta, May 2010. ELRA.

[10]TensorFlow. Word2vec. `https://www.tensorflow.org/tutorials/text/word2vec,2021`

[11]`https://github.com/akoksal/Turkish-Word2Vec`

[12]`https://github.com/graykode/nlp-tutorial`

[13]`https://nlp.stanford.edu/projects/glove/`

words, FastText represents the words as a bag of character n-grams. Words are represented as the sum of these representations. FastText is a fast method, having a reasonable amount of training time on large corpora and enabling computing word representations for words that did not appear in the training data. FastText library is developed by Facebook AI Research and publicly available[14] .

Later on, context-dependent embedding methods are developed such as BERT and Elmo. BERT (Bidirectional Encoder Representations from Transformer) is introduced in the paper *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* [5] by Devlin et. al. As stated by the Google AI Language, "BERT outperforms previous methods because it is the first unsupervised, deeply bidirectional system for pre-training NLP"[15] . BERT has been created to pretrain intricate, two-way representations from text that has not been labeled, by conditioning on both the context to the left and right in every layer.

Lastly, in their paper *Deep contextualized word representations* [6] by Allen Institute for Artificial Intelligence, and University of Washington, a new approach to deep contextualized word representation is introduced that incorporates two distinct aspects: the intricate attributes of word utilization, such as syntax and semantics, and the dynamic variations of these uses in various linguistic contexts, thereby modeling polysemy. PyTorch[16] and TensorFlow[17] implementations of biLM (deep bidirectional language model) to compute ELMo representations can be found online. In our work, we have used Turkish CoNLL17 corpus [9] trained with ELMo and used ELMoForManyLanguages [10] repository to use Turkish word embeddings with API.

4. Methodology

In this section we will be explaining the mathematical concepts behind the state-of-the-art techniques used for the Turkish word embedding repository.

4.1. Word2Vec

Word2Vec is a popular deep learning-based method for generating word representations or word embeddings. It's a two-layer neural network that takes a large corpus of text as input and learns to represent each word in a high-dimensional vector. The basic idea behind Word2Vec is that words that appear in similar contexts in the corpus should have similar representations. This is achieved by training the neural network to predict the surrounding words of a target word given its representation. There are two main variants of Word2Vec: CBOW (Continuous Bag of Words) and Skip-gram. CBOW tries to predict a target word based on the surrounding words, while Skip-gram tries to predict the surrounding words based on the target word. In the second paper by Mikolov et. al. [1], Skip-gram model is improved using *Negative Sampling* and *Hierarchical Softmax*.

Skip-gram Word2Vec architecture consists of an input layer, an output layer, and a single hidden layer. The input layer represents the one-hot encoding of the target word w_t, and the output layer is a dense layer with a *softmax activation function* that predicts the proba-

[14] https://github.com/facebookresearch/fastText
[15] https://github.com/google-research/bert
[16] https://github.com/allenai/allennlp/blob/main/allennlp/modules/elmo.py
[17] https://github.com/allenai/bilm-tf

bility distribution over the context words. There can be more than one such layers in *parallel*, representing probability distributions for context words w_{t+j} for different values of j.

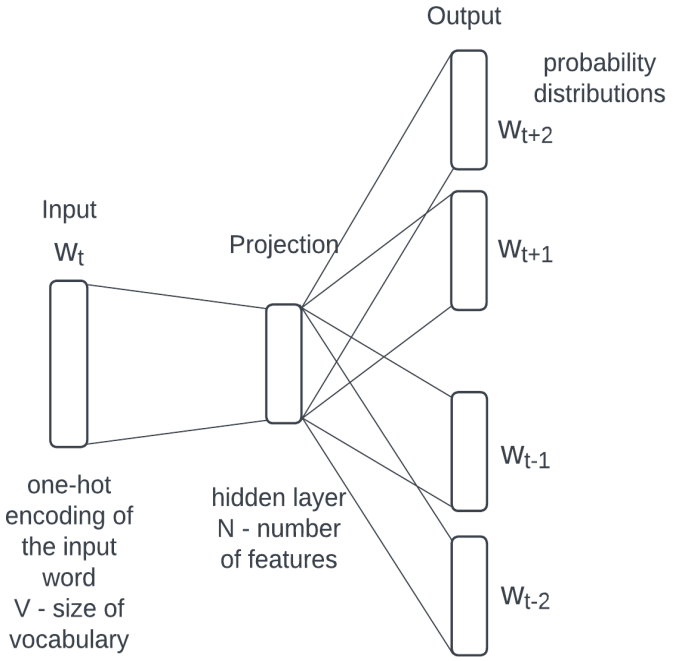

Figure 4.1: The Skip-gram model architecture for window size = 2 [2].

The objective of the Skip-gram model is to maximize the average log-likelihood of the observed words given the surrounding context words in a corpus, which is formulated as:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log p(w_{t+j}|w_t) \tag{4.1}$$

m is the size of the training context and T stands for number of training words. Similar to other loss functions generally used in Machine Learning, we prefer to represent the loss function with logarithm. Log probabilities are crucial in digital probability for various reasons. One reason is that computers have limitations in representing extremely small numbers. Additionally, logs possess a remarkable ability to convert multiplication into addition, and computers can perform addition much faster[18] . Probability of a surrounding word w_{t+j} given target word w_t, $p(w_{t+j}|w_t)$

[18]Piech, C. (n.d.). Log probabilities. In Probability for Computer Scientists. Retrieved from `https://chrispiech.github.io/probabilityForComputerScientists/en/part1/log_probabilities/`

is defined as:

$$p\left(w_{t+j} \mid w_t\right)=\frac{\exp\left(\mathrm{v}_{w_{t+j}}^T \mathrm{u}_{w_t}\right)}{\sum_w^V \exp\left(\mathrm{v}_w^T \mathrm{u}_{w_t}\right)} \tag{4.2}$$

where $\mathrm{v}_{w_{t+j}}$ and u_{w_t} are the word embeddings for the context word w_{t+j} and target word w_t, respectively, and V is the vocabulary size. u represents input representation of w, whereas v represents output representation in $\mathbb{R}^d$. Due to the denominator, cost of calculating the given probability is proportional to the vocabulary size V, which is often large (10^5 - 10^7 terms) [2].

Hierarchical softmax was first introduced by Morin and Bengio (2005) [11] in neural probabilistic language models. The approach is founded on an idea that has the potential to offer near-exponential acceleration in terms of the number of words present in the vocabulary. It replaces the normalization term with a binary tree structure, where the root node represents the entire vocabulary, and the leaves represent individual words. By constructing an appropriate tree based on a corpus, the complexity of computing softmax can be reduced from $O(V)$ to $O(log(V))$, where V is the corpus size. For instance, if there are 10,000 words in the corpus, a two-layer hierarchical softmax can be created, with each node in the first layer having 100 child nodes, and each node in the second layer having 100 child nodes as well. With conventional softmax, output of the activation function would need to be computed 10,000 times. With hierarchical softmax, however, output would only need to be computed 100 times in the first layer and then another 100 times for the second layer, resulting in a total of 200 computations[19] .

The hierarchical softmax employs a binary tree structure to represent the output layer, with the V words serving as its leaves. Each node in the tree explicitly denotes the probabilities of its child nodes. These probabilities create a random path that allocates probabilities to words. Let w_I to be the input word and w_O to be the output word. $L(w_O)$ is the length of the path from the root to w_O in the tree. $n(w, j)$ is the j-th node on the path from the root to w and $ch(n)$ is an arbitrarily fixed child of node n. Associate vectors v_n with each node n in the tree. $[\![x]\!]$ is 1 if x is true, otherwise -1. Then the hierarchical softmax can be formulated as follows [2]:

$$p(w_O|w_I)=\prod_{j=1}^{L(w_O)-1} \sigma\left([\![n(w_O, j+1)=ch(n(w_O, j))]\!] \cdot v_{n(w_O,j)}^T \cdot u_{w_I}\right) \tag{4.3}$$

where $\sigma=(1+e^{-x})^{-1}$. Another method used for the skip-gram model is Noise Contrastive Estimation (NCE). The basic idea behind NCE is to use a set of *negative samples*, drawn from a noise distribution, to estimate the likelihood of positive samples in a dataset. By using negative samples, we eliminate the need to go over the whole vocabulary. This significantly speeds up the training process and allows for larger vocabulary sizes. $p(w_{t+j}|w_t)$ is replaced by the following

[19]Mao, Lei. "Hierarchical Softmax." *Lei Mao's Log Book*. 2018. Available at: `https://leimao.github.io/article/Hierarchical-Softmax/`. Accessed on: Feb 25, 2023.

equation for negative sampling:

$$= \frac{1}{T}\sum_{t=1}^{T}\left[\log\sigma(\mathrm{v}_{w_O}^{T}\mathrm{u}_{w_I}) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w)}\left[\log\sigma(-\mathrm{v}_{w_i}^{T}\mathrm{u}_{w_I})\right]\right] \tag{4.4}$$

where k stands for the number of negative samples for training. Experimentally, it has been found that values of k between 5 and 20 tend to be effective for smaller training datasets, while for larger datasets, values as low as 2 to 5 can suffice [2]. To get negative samples, Mikolov et. al. experimentally found out that the best-performing distribution for $P_n(w)$ is the unigram distribution $U(w)$ raised to the $\frac{3}{4}$rd power, which happened to outperform the unigram and the uniform distributions.

Lastly, Mikolov et. al. points out the negative effects of frequently reoccurring words such as "the", "a", "an" in English. A method called subsampling of frequent words is used to get rid of redundant words in the corpus. To eliminate the imbalance between rare and frequent words, each word w_i in the training set is removed with the probability computed by the given expression:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{4.5}$$

where $f(w_i)$ is the frequency of the word w_i and t is the chosen threshold. It is stated in the paper [2] that although this subsampling formula was chosen heuristically, it is found to be working pretty well in practice.

We utilized two different implementations for the Word2Vec architecture in our study. To start, we used the straightforward Word2Vec implementation from the TensorFlow Core Tutorials [20] as an introductory tool for gaining familiarity with the concepts. Later on, we employed the Gensim library [7], which is a free and open-source Python library for representing documents as semantic vectors, for larger datasets. The second implementation was preferred for its optimized and well-tested architecture. Gensim library provides the opportunity to experiment with various algorithms such as Skip-Gram with hierarchical softmax, Skip-Gram with negative sampling, CBOW with hierarchical softmax, and CBOW with negative sampling.

4.2. GloVe

GloVe is a word embeddings method which is developed based on analysis of model properties needed semantic and syntactic regularities to emerge in word vectors. It aims to explicitly explicitly embed these properties into the model structure. The end result is a global log-bilinear regression model which incorporates the advantages of two prominent model families in natural language processing literature: global matrix factorization (such as latent space analysis of

[20] TensorFlow. Word2vec. `https://www.tensorflow.org/tutorials/text/word2vec,2021`

Deerwester et al., 1990 [12]) and local context window (such as skip-gram model of Mikolov et al., 2013 [13]). The resulting GloVe model achieves performance improvements on the analogy, similarity and named entity recognition (NER) benchmarks.

The GloVe model is based on an analysis of why statistics of word occurrences in a corpus is the primary source of information used by unsupervised methods for learning word representations. Let X denote word-word co-occurrence counts whose entries $X_{i,j}$ are the number of times word j occurs in the context of word i. Let $X_i = \Sigma_k X_{ik}$ be the number of times any word appears in the context of word i. Let $P_{ij} = P(j|i) = X_{ij}/X_i$ be the probability that word j appears in the context of word i.

Now consider the two related words $i = ice$ and $j = steam$; and probe words $k_0 = solid$, $k_1 = water$ and $k_2 = fashion$. For k_i which is related to ice but unrelated to $steam$, we would expect the ratio P_{ik}/P_{jk} to be large. For k_i which is related to both ice and $steam$ or neither, we would expect the ratio P_{ik}/P_{jk} to be close to 1. Compared to the raw probabilities P_{ij}, probability ratio is a better indicator of relevance of the words and it is better at discriminating between two similar words. This argument suggests that the starting point for a word vector learning algorithm should be with the ratios of co-occurrence probabilities rather than the probabilities themselves. The most general model takes the form:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{4.6}$$

where $w \in \mathbb{R}^d$ are word vectors and $\tilde{w} \in \mathbb{R}^d$ are context words. Right hand side of this equation can be extracted from the corpus. Possibilities for F are countless but a unique choice can be made by enforcing some conditions.

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{4.7}$$

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{4.8}$$

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{4.9}$$

In equation (4.8); two parameters w_i and w_j are transformed to a single parameter $w_i - w_j$ using vector difference since vector spaces are inherently linear structures. Then, in equation (4.9), dot product is applied to the vector arguments of F to match the right hand side of the equation which is a scalar. Next, homomorphism is assumed between the groups $(\mathbb{R},+)$ and $(\mathbb{R}_{>0},\times)$. This assumption transforms the left hand side of (4.9) to:

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} \tag{4.10}$$

which, by equation (4.9), is solved by:

$$F(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i} \tag{4.11}$$

If F is selected to be the exponential function (which satisfies the homomorphism assumption), equation (4.11) becomes:

$$w_i^T \tilde{w}_k = log(P_{ik}) = log(X_{ik}) - log(X_i) \tag{4.12}$$

This equation is simplified further by absorbing the $log(X_i)$ into the bias of w_i, b_i. Bias of $\tilde{w}_k$ is also added to restore symmetry:

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = log(X_{ik}) \tag{4.13}$$

After adding a weight factor which gives more weight to the term with higher frequency in the corpus, loss function is written with equation (4.13):

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_k + b_i + \tilde{b}_k - log(X_{ik}))^2 \tag{4.14}$$

where f is the weight function and V is the size of the vocabulary.

Using this log-bilinear regression model, word representations that outperform previous methods on word analogy, word similarity, and named entity recognition tasks can be trained.

4.3. FastText

FastText, developed by Facebook's AI Research team (Joulin et al., 2016) [4], operates on the belief that the form of a word holds significant information about its meaning. Unlike traditional word embeddings like Word2Vec, which create a separate embedding for each distinct word, FastText takes the morphological structure into account. This is especially relevant in languages with a rich morphological structure such as Turkish or Japanese, where a single word can have numerous forms, many of which may be infrequent. This can make it challenging to generate effective word embeddings. In contrast to Word2Vec, FastText allows us to calculate word embeddings for words that were not present in the training dataset as well.

In the FastText algorithm, each word w is represented by character n-grams. To create the n-grams for a word, special symbols are appended to the beginning and end of it. For $n = 4$ and the word *world*, following character n-grams are generated:

$$< \text{wor}, \quad \text{worl}, \quad \text{orld}, \quad \text{rld} >$$

According to Joulin et al., 2016 [4], usually all the n-grams for $3 \leq n \leq 6$ are extracted for training. Additionally, the word w itself is also added in the set of its n-grams. FastText represents the probability of a context word w_{t+j} given the target word w_t, denoted as $p(w_{t+j}|w_t)$, in a way similar to Word2Vec, but with some modifications. Specifically, FastText introduces a

score function s and formulates the probability as follows:

$$p\left(w_{t+j} \mid w_t\right) = \frac{e^{s(w_t, w_{t+j})}}{\sum_{i=1}^{V} e^{s(w_t, i)}} \tag{4.15}$$

In Word2Vec, $s(w_t, w_{t+j})$ was equal to $\mathrm{v}_{w_{t+j}}^T \mathrm{u}_{w_t}$. Joulin et al., 2016 [4] proceeds with a dictionary n-grams of size G. For any word w, $\mathcal{G}_w \subset \{1, ..., G\}$ denotes the n-grams appearing in word w. Each n-gram g is associated with a vector z_g. Then the scoring function can be defined as the sum of the n-gram vector representations of word w as follows:

$$s(w_t, w_{t+j}) = \sum_{g \in \mathcal{G}_w} {z_g}^T v_{w_{t+j}} \tag{4.16}$$

Using this approach, FastText is able to calculate word embeddings for words that were not present in the vocabulary - but can be constructed using different n-grams. During the training, Optimization is performed using stochastic gradient descent on the negative log-likelihood presented before. Implementation of this algorithm by Facebook AI can be found on Github[21] .

4.4. ELMo

ELMo [6], also known as "Embeddings from Language Model," is a technique used for generating word embeddings. It represents a series of words as a corresponding sequence of vectors. Unlike "Bag of Words" methods and earlier vector approaches such as Word2Vec and GloVe, ELMo embeddings are context-sensitive, producing distinct representations for words that share the same spelling but have different meanings, such as "bank" in "river bank" and "bank balance". To produce word-level embeddings, bi-directional LSTM receives character-level tokens as inputs. This approach is similar to our previous model BERT in terms of its *contextual sensitivity*[22] .

LSTM (Long-short Term Memory), first developed by Hochreiter et. al. [14], is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs [15]. LSTM successfully deals with some of the problems such as exploding/vanishing gradients RNNs might encounter within various tasks. In a many-to-many model, an LSTM network maps input sequence $(x_1, x_2, ..., x_t)$ to $(y_1, y_2, ..., y_t)$ by forwarding hidden and cell states using the equations

[21] `https://github.com/facebookresearch/fastText`
[22] `https://en.wikipedia.org/wiki/ELMo`

provided below from $t = 1$ to T:

$$\Gamma_i^{<t>} = \sigma(W_{ix}x^{<t>} + W_{ia}a^{<t-1>} + b_i) \tag{4.17}$$

$$\Gamma_f^{<t>} = \sigma(W_{fx}x^{<t>} + W_{fa}a^{<t-1>} + b_f) \tag{4.18}$$

$$c^{<t>} = \Gamma_f^{<t>} \odot c^{<t-1>} + \Gamma_i^{<t>} \odot g(W_{cx}x^{<t>} + W_{ca}a^{<t-1>} + b_c) \tag{4.19}$$

$$\Gamma_o^{<t>} = \sigma(W_{ox}x^{<t>} + W_{oa}a^{<t-1>} + b_o) \tag{4.20}$$

$$a^{<t>} = \Gamma_o^{<t>} \odot h\left(c^{<t>}\right) \tag{4.21}$$

$$y^{<t>} = \phi(W_{ya}a^{<t>} + b_y) \tag{4.22}$$

The model architecture [15][23] includes several weight matrices denoted by the W terms (for example, W_{ix} represents the weight matrix connecting the input gate to the input). Additionally, there are diagonal weight matrices for peephole connections denoted by W_{ic}, W_f, and W_{oc}. The bias vectors are denoted by the b terms (such as b_i for the input gate bias vector). The logistic sigmoid function σ is applied to the input gate, forget gate, output gate, and cell activation vectors Γ_i, Γ_f, Γ_o, and c respectively, which are all of the same sizes as the cell output activation vector h. The element-wise product of vectors is represented by $\odot$. The cell input and cell output activation functions are represented by g and h, which are generally the hyperbolic tangent function ($tanh$). Finally, the network output activation function is denoted by ϕ and is typically softmax. Below you can find the inner architecture of an LSTM cell[24] .

Given a sequence of N tokens, a forward language model computes the probability of the sequence using conditional probability as follows:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1}) \tag{4.23}$$

In a classical LSTM architecture with L layers forward, a token embedding x_k^{LM} is passed through the layers. For creating the token embeddings, different architectures can be used such as character-level CNNs[25] . At each position k, a context-dependent representation $\overrightarrow{h}_{k,j}^{LM}$ is generated. Here j represents the layers from 1 to L and the right arrow indicates that the flow is from the leftmost word to the rightmost word in the sentence. In the top layer, $\overrightarrow{h}_{k,L}^{LM}$ is fed to a softmax layer to predict the token t_{k+1}. Using this approach, we are able to utilize all the words t_1 to t_k to predict $t_k + 1$.

[23]In the original paper *Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling*, there are extra $W_{oc}c^{<t>}$ terms in 4.8, 4.9 and 4.11. Later on, these parts are removed for computational efficiency.

[24]Retrieved from `https://www.coursera.org/specializations/deep-learning`.

[25]`https://github.com/allenai/allennlp/blob/main/allennlp/modules/elmo.py`

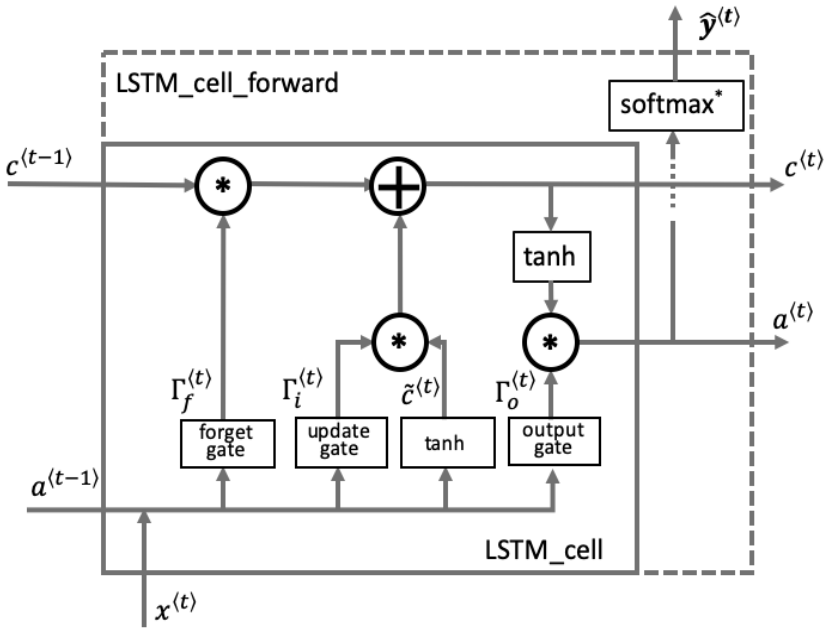

Figure 4.2: Inside architecture of an LSTM cell

In bi-directional LSTM (which is used by biLM), sequence information in both directions backward (future to past) and forward (past to future) are processed. Backward iteration attempts to predict the previous token given the future context:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \ldots, t_N) \tag{4.24}$$

Using a bi-directional LSTM, our task reduces to maximizing the log-likelihood of the predictions using both left and right context:

$$\sum_{k=1}^{N} \left(\log p \left(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s \right) + \log p \left(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s \right) \right) \tag{4.25}$$

Here, Θ_x represents the parameters used for token representation, Θ_s represents the parameters in dense layer used just before the softmax activation function. Lastly, Θ_{LSTM} represents the parameters used in the forward and backward directions. In ELMo, some of the weights are shared between forward and backward directions. Overall, for an L-layer bidirectional LSTM, $2L-1$ vectors are computed: x_k^{LM}, $\overrightarrow{h}_{k,L}^{LM}$, $\overleftarrow{h}_{k,L}^{LM}$. In the simplest case, we can select the outputs of the top layer. However, in general, in ELMo it is preferred to compute a task-specific weighting

of all biLM layers:

$$ELMo_k^{task} = E(R_k; \Theta^{task}) = \gamma_{task} \sum_{j=0}^{L} s_j^{task} \cdot h_{k,j}^{LM} \tag{4.26}$$

Here, s^{task} is the softmax normalized weights and γ_{task} is used to rescale the whole vector (practically useful for optimization). It is also stated by E. Peters et. al. it is beneficial to add a moderate amount of dropout to ELMo and regularize it by adding $\lambda \|\mathbf{w}\|_2^2$ to the loss function.

5. Requirements and Modeling

Our project's primary objective was to create word embedding repositories for Turkish using existing model implementations. As such, traditional software requirements were not applicable in this context. However, we did develop a model evaluation package, which we will elaborate on in the Requirements and Modelling section. This package allows for the loading of different models, performing analogy and similarity tasks, saving results, and generating LaTeX tables.

5.1. System Requirements

5.1.1. User Requirements

- **5.1.1.1:** Users shall be able to associate reference scores with groups of analogy or similarity tasks.
- **5.1.1.2:** Users shall be able to load any type of pre-trained model for generating word embeddings.
- **5.1.1.3:** Users shall be able to run analogy or similarity tasks with the loaded models.
- **5.1.1.4:** Results of the analogy and similarity tasks shall be saved in a file format that is easily accessible by users.
- **5.1.1.5:** Users shall be able to generate latex tables from the task results and the reference scores.

5.1.2. System Requirements

- **5.1.2.1:** Package shall offer a CLI when loading models to make it easier to configure the way model is loaded.
- **5.1.2.2:** The package should be able to handle large datasets and perform the analogy and similarity tasks in a reasonable amount of time.
- **5.1.2.3:** The package should be maintainable and easy to update in the event that new models or tasks are added in the future.

5.2. System Modeling

The evaluation package is composed of several modules that interact with one another to provide the desired functionality. The main entry point for users is a set of four scripts that

allow them to load models, run analogy and similarity tasks, and save the results in various formats. The first script, "evaluate", prompts the user to configure the way the model is loaded and then runs the tasks provided by the user. The second script, "populate_metadata", allows the user to associate a reference score with a given task. Finally, the "dump_latex_table" script allows the user to generate a LaTeX table with the score and reference score of a given task.

Under the hood, the evaluation package relies on several modules for loading models and performing tasks. When the "evaluate" script is run, it uses a model loader module to read the specified model path and load the pre-trained model into memory. The script then uses a task runner module to execute the provided tasks against the loaded model and save the results in a human-readable format. The "populate_metadata" script relies on a metadata loader module to read in the reference scores for each task and store them in memory. Finally, the "dump_similarity_latex_table" and "dump_analogy_latex_table" scripts uses a table generator module to convert the saved results into a LaTeX table format.

5.3. System Architecture

The evaluation package is designed to be modular and extensible, with a clear separation of concerns between the different components. The system architecture consists of several main modules, each responsible for a specific aspect of the system's functionality.

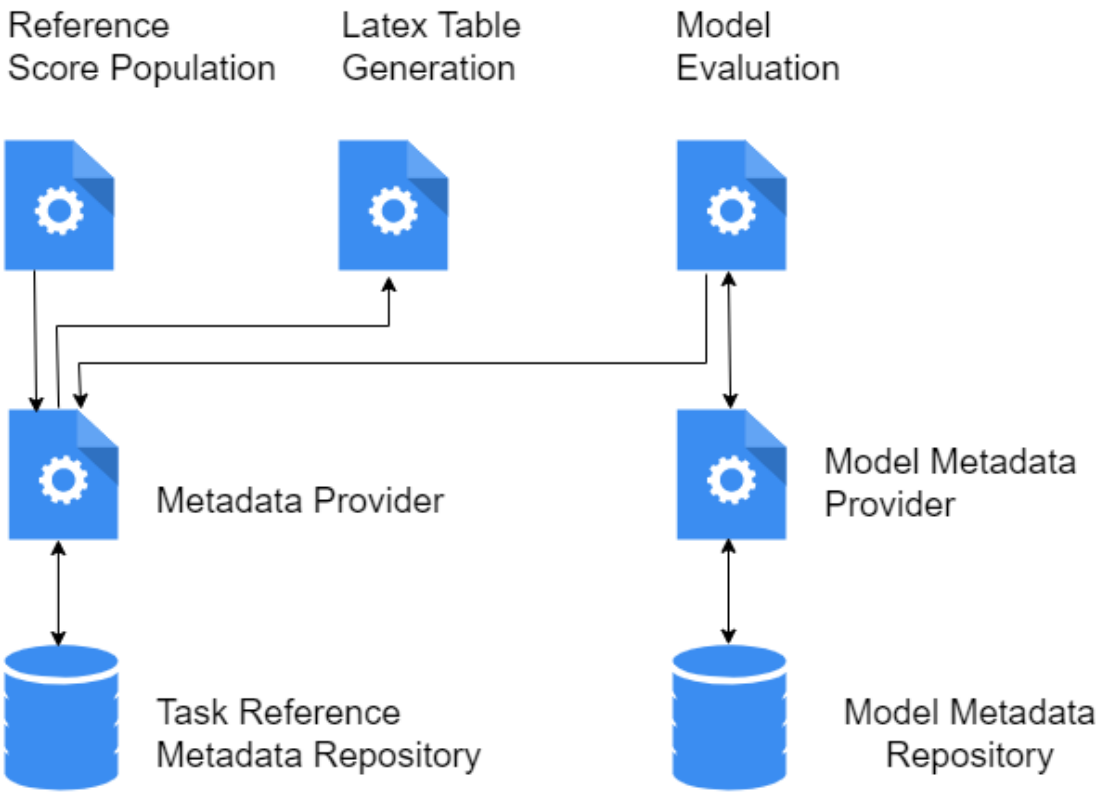

Figure 5.1: System Model Diagram

At the core of the system is the model loader module, which is responsible for reading in pre-trained models and loading them into memory. This module provides a common interface for loading different types of models and ensures that the loaded models are compatible with the rest of the system.

The task runner module is responsible for executing the various analogy and similarity tasks against the loaded model. It provides a flexible framework for defining new tasks and allows users to easily specify the tasks they wish to run via the "evaluate" script.

The metadata loader module is responsible for reading in reference scores for each task and storing them in memory. This module allows users to provide ground truth scores for each task and enables the system to generate accurate evaluation metrics.

Finally, the table generator scripts are responsible for converting the saved results into a LaTeX table format. These scripts provide a convenient way for users to generate tables summarizing the results of their evaluations.

Overall, the system architecture is designed to be flexible and modular, with each component responsible for a specific aspect of the system's functionality. The clear separation of concerns between the different modules allows for easy extensibility and maintainability of the system.

6. Design, Implementation, and Testing

6.1. Design & Implementation

Our software package can be split to two groups, first group being the evaluation package we described in the Requirements and Modeling section and second group being model specific scripts we prepared for training models.

In the evaluation package, we began with implementing the task and model metadata repository and providers. Task metadata provider was implemented under the `MetaData` class. This class allows one to load existing metadata, update results and save them. Then, using the `MetaData` class, we wrote the `populate\_task\_metadata.py` script which is a CLI tool for saving reference scores for tasks. Task metadata is saved in the folder where the tasks are as a csv file whose name ends with `.nlp\_metadata`.

Then we wrote the model metadata provider under the `Model\_MetaData` class. This class allows one to load and save model metadata. Using `Model\_MetaData` and `MetaData` classes, we implemented the `evaluate.py` script which takes a model path, list of tasks to run as parameter and runs the provided tasks with the provided model. If the user is loading a particular model for a first time, they are prompted to configure how the model is to be loaded. After the loading is configured, the configuration is dumped as a JSON file whose name ends with `.nlp\_metadata`.

When loading and running the models, we use the *Gensim* library which allows us to represent the models as a dictionary mapping words to word embedding vectors.

Finally, we added the `dump\_analogy\_latex\_table.py` and `dump\_similarity\_latex\_table.py` scripts to generate latex tables from the results of our experiments. Together with the other scripts which allowed us to run experiments quickly, these scripts allowed us to generate several tables in quick succession.

For the model specific scripts, we have mainly written regular Python files (.py) and Jupyter notebooks (.ipynb). We have maintained a Github repository[26] consisting of different word embedding model implementations. Each model has its own folder. There are also several preprocessing scripts to manipulate our datasets before training our models on them. We have used *Gensim* library for Word2Vec and FastText models. For GloVe, we used the official implementation [27] for training the model and gensim for loading.

6.2. Testing

We have used several intrinsic evaluation methodologies and metrics to test the quality of the generated word embeddings. Intrinsic evaluation involves evaluating the quality of word embeddings on a specific task that is designed to test a particular linguistic property. For example, word similarity tasks, such as the WordSim-353 and SimLex-999 datasets, can be used to evaluate how well the embeddings capture the semantic similarity between pairs of words. *Analogy tasks* and *similarity tasks* are two types of intrinsic evaluation methodologies that are commonly used to test the quality of word embeddings.

In analogy tasks, the performance of word embeddings is evaluated on their ability to solve analogy problems, such as "*man* is to *king* as *woman* is to *queen*". This classic analogy can be solved using word embeddings by finding that closest to $w_{\text{king}} - w_{\text{man}} + w_{\text{woman}}$, which turns out to be w_{queen} (w_x denotes the embedding of word x). In geometric terms, word embeddings of analogies approximately form parallelograms [28] .

For the analogy task, we have used the metric MRR (Mean Reciprocal Rate) used by Güngör et. al. on Turkish word embeddings [8]. The mean reciprocal rank (MRR) is a statistical measure used to evaluate systems that generate lists of possible responses to a set of queries, sorted by their probability of being correct [29] . The reciprocal rank of a query response is the inverse of its rank in the list of correct answers, with a value of 1 for the first correct answer, $\frac{1}{2}$ for the second, $\frac{1}{3}$ for the third, and so on. It can be expressed as follows:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{S_i} \tag{6.1}$$

where Q represents the set of analogy queries, and S_i indicates the position of the correct answer in the list of closest words. In the paper *Efficient Estimation of Word Representations in Vector Space*, a slightly different approach is used where the "question is assumed to be correctly answered only if the closest word to the vector computed using the above method is exactly the same as the correct word in the question".

We have tested our models on the analogy dataset used by Güngör et. al., and compared our results in the next chapter. Specifically, we have used the syntactic features *isim çekim ekleri*

[26] https://github.com/Turkish-Word-Embeddings/Word-Embeddings-Repository-for-Turkish
[27] https://github.com/stanfordnlp/GloVe
[28] https://carl-allen.github.io/nlp/2019/07/01/explaining-analogies-explained.html
[29] https://en.wikipedia.org/wiki/Mean_reciprocal_rank

and *fiil çekim ekleri*. The divided version of the dataset can be found in our Github repository[30] . Additionally, we utilized another dataset that can be found online [31] for both semantic and analogy tasks.

In natural language processing, similarity task for word embeddings refers to measuring the degree of similarity or relatedness between two or more words based on their vector representations obtained from a pre-trained word embedding model. For such purposes, we have used *WordSimTr* dataset prepared by Üstün et al. [16] for Turkish word embedding models. We utilized 140 word pairs that scored between 1 to 10. Our objective is to compute similarity scores for these word pairs using our model, and then compare the outcomes with hypothetical ones. The outcome of this comparison can be conveyed using either the *Pearson Correlation* or *Spearman Correlation* coefficients. In our example, Pearson Correlation between two samples can be defined as follows:

$$P = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{6.2}$$

where x_i and y_i represent the similarity scores for each word pair. Similarly, Spearman Correlation between two samples can be defined as follows:

$$S = 1 - \frac{6\sum_{i=1}^{n}(R(x_i) - R(y_i))^2}{n(n^2 - 1)} \tag{6.3}$$

where n is the number of scores in the dataset and R represents the ranking. In statistics, ranking is the data transformation in which numerical or ordinal values are replaced by their rank when the data are sorted. For example, the numerical data 3.4, 5.1, 2.6, 7.3 are observed, the ranks of these data items would be 2, 3, 1 and 4 respectively[32] . S is computed on ranks and so depicts *monotonic* relationships while P is on true values and depicts linear relationships[33] . A monotonic relationship is one in which the size of one variable increases as the other variables also increase (positive monotonic), or where the size of one variable increases as the other variable also decreases (negative monotonic). Finally, the p-value can be defined as the probability of observing a correlation between two sample sets by chance if no correlation truly exists. The resulting coefficients are considered to be more statistically significant when the p-value is low.

7. Results

We have trained our models with the following configurations. We have used the paper

[30] https://github.com/Turkish-Word-Embeddings/Word-Embeddings-Repository-for-Turkish/tree/main/tasks/analogy

[31] https://github.com/bunyamink/word-embedding-models/tree/master/datasets

[32] https://en.wikipedia.org/wiki/Ranking#Ranking_in_statistics

[33] https://stats.stackexchange.com/q/14963

Linguistic Features in Turkish Word Representations by Güngör et. al. as a reference for MRR results. Overall weighted results are calculated based on the number of examples.

We have used the Turkish word embeddings generated by FastText [17] as our reference and compared our results with it. We have trained our FastText model using Gensim with the following configurations.

- As our dataset, we used the merged version of *BounWebCorpus*[34] and *HuaweiCorpus*[35] . Final training corpus has a size of ∼10.5 GB. Overall, we had 1,384,961,747 tokens and 1,573,013 unique words (excluding words occurring less than the minimum frequency).
- For our models, number of embedding features was set to 300.
- Window (maximum distance between the current and predicted word within a sentence) size was set to 5.
- The number of negative samples for negative sampling was set to 5.
- The minimum frequency was set to 10.
- The number of iterations (epochs) over the corpus was set to 5 and 10 respectively.
- For the analogy task, N was set to 10.
- *Skip-gram* with *negative sampling* was used.
- (Relevant for FastText) Minimum length of a char gram was set to 3, and the maximum length was set to 6. The maximum length of word gram was set to 1.
- (FastText Turkish Word Embeddings [17]) Turkish word embeddings were trained using CBOW with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5 and 10 negatives.
- For ELMo, we have used Turkish CoNLL17 corpus[36] prepared by Language Technology Group at the University of Oslo. This model had 327,299 unique words in total.

In the following table, you can find the overall performance of the models we evaluated in our experiments.

[34] https://tulap.cmpe.boun.edu.tr/repository/xmlui/handle/20.500.12913/16
[35] https://github.com/onurgu/linguistic-features-in-turkish-word-representations/releases
[36] http://vectors.nlpl.eu/repository/

#	Similarity				Analogy			
	Syntactic		Semantic		Fiil Çekim Ekleri	İsim Çekim Ekleri	Semantic	Yapım Ekleri
	Pearson	Spearman	Pearson	Spearman				
Word2Vec 5 epochs	**83.80**	**82.57**	70.72	74.33	0.604	0.318	0.535	-
Word2Vec 10 epochs	82.53	81.68	**70.77**	**74.57**	0.609	0.329	0.551	-
Word2Vec 10 epochs left-aligned	79.05	79.42	69.85	73.98	0.287	0.236	0.535	
Word2Vec 10 epochs right aligned	79.02	76.72	70.11	73.17	0.551	0.266	0.463	
Word2Vec 15 epochs	81.54	81.83	70.67	74.53	0.606	0.337	**0.555**	
FastText 5 epochs	61.12	62.55	67.45	70.37	0.718	0.360	0.258	
FastText 10 epochs	66.51	68.16	67.35	70.53	0.697	0.371	0.303	
FastText Facebook	76.92	80.28	65.12	68.92	**0.801**	**0.514**	0.346	
Fasttext and Word2vec Average	79.12	79.19	68.15	71.98	0.658	0.368	0.470	
GloVe window size 5	73.45	74.13	61.59	64.33	0.549	0.144	0.460	
GloVe window size 10	69.87	72.33	67.93	70.89	0.363	0.171	0.501	
Decontextualized Elmo	39.11	39.43	35.18	36.77	0.237	0.162	0.078	

You can find the results of CBOW-Skipgram Word2Vec comparison, Word2Vec with 5 epochs, Word2Vec with 10 epochs, left-aligned Word2Vec with 10 epochs, right-aligned Word2Vec with 10 epochs, Word2Vec with 15 epochs, FastText with 5 epochs, FastText with 10 epochs, FastText trained by Facebook AI, average of the word vectors generated by Word2Vec and FastText with 10 epochs, GloVe with 100 iterations and ELMo (Turkish CoNLL17) on static embeddings in the following tables.

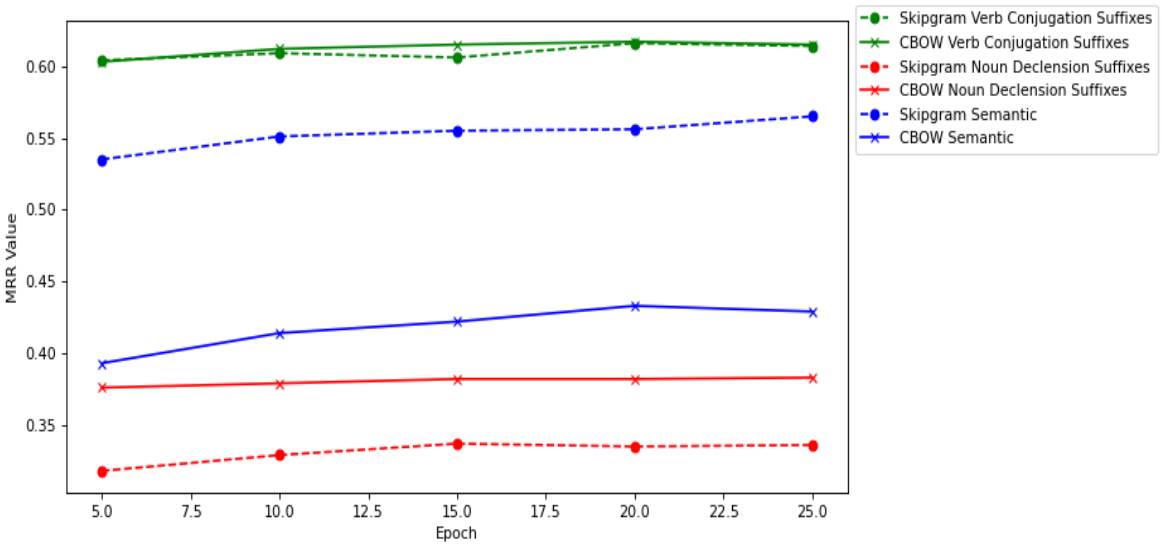

Figure 7.1: Comparing the CBOW and Skip-gram architectures on analogy tasks, we find that they perform similarly on verb conjugation suffixes. However, when it comes to noun declension suffixes, CBOW outperforms Skip-gram. On semantic analogy tasks, Skip-gram performs better than CBOW.

Morphological Categories	Number of examples	Top-N Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.288	0.439	0.339	29.68%
2. tekil kişi çekimi.txt	946	0.411	0.323	0.365	-11.48%
2. çoğul kişi çekimi.txt	946	0.349	0.398	0.327	21.94%
3. çoğul kişi çekimi.txt	1128	0.419	0.372	0.42	-11.22%
emir kipi.txt	1176	0.187	0.632	0.465	36.06%
gelecek zaman kipi.txt	1176	0.105	0.801	0.707	13.37%
gereklilik kipi.txt	1128	0.232	0.561	0.467	20.19%
geçmiş zaman eki (-di).txt	1176	0.225	0.622	0.535	16.25%
geçmiş zaman eki (-miş).txt	1176	0.088	0.799	0.774	3.32%
istek kipi.txt	1176	0.249	0.582	0.534	8.97%
olumsuzluk eki.txt	1176	0.137	0.642	0.67	-4.08%
şimdiki zaman eki (-mekte).txt	1176	0.132	0.749	0.646	15.99%
şimdiki zaman eki (-yor).txt	861	0.077	0.874	0.847	3.26%
Overall	14231	0.220	0.604	0.549	10.02%

Table 7.1: *Fiil çekim ekleri*: MRR Result Comparison for **Word2Vec** with 5 epochs

Semantic Categories	Number of examples	Top-N Miss Ratio	MRR
Aile	90	0.289	0.526
Para birimi	156	0.730	0.149
Şehir-bölge	1344	0.092	0.605
Ülke-başkent	506	0.081	0.656
Eş anlamlılar	600	0.4	0.467
Zıt anlamlılar	600	0.336	0.448
Overall	3296	0.226	0.535

Table 7.2: *Semantic categories*: MRR Results of **Word2Vec** with 5 epochs

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 83.80	p-value: 6.83×10^{-21}
	Spearman Result: 82.57	p-value: 7.98×10^{-20}
	OOV Ratio: 46.42	
Semantic Similarity	Pearson Result: 70.72	p-value: 6.38×10^{-57}
	Spearman Result: 74.33	p-value: 9.93×10^{-66}
	OOV Ratio: 26.6	

Table 7.3: Similarity test results for **Word2Vec** with 5 epochs ($p \times 100$)

Morphological Categories	Number of examples	Top-N Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
Şimdiki zaman eki (-yor)	861	0.067	0.884	0.847	4.37%
geçmiş zaman eki (-miş)	1176	0.086	0.808	0.774	4.39%
Gelecek zaman eki	1176	0.109	0.792	0.707	12.05%
Olumsuzluk eki	1176	0.136	0.651	0.67	-2.83%
Şimdiki zaman eki (-mekte)	1176	0.115	0.760	0.646	17.65%
Geçmiş zaman eki (-di)	1176	0.220	0.633	0.535	18.32%
İstek kipi	1176	0.237	0.593	0.534	11.02%
Gereklilik kipi	1128	0.238	0.575	0.467	23.10%
Emir kipi	1176	0.193	0.643	0.465	38.29%
3. çoğul kişi çekimi	1128	0.392	0.388	0.42	-7.61%
2. tekil kişi çekimi	946	0.426	0.315	0.365	-13.70%
1. tekil kişi çekimi	990	0.311	0.426	0.339	25.66%
2. çoğul kişi çekimi	946	0.358	0.380	0.327	16.20%
Overall	14231	0.219	0.609	0.549	10.96%

Table 7.4: *Fiil çekim ekleri*: MRR Result Comparison for **Word2Vec** with 10 epochs

Morphological Categories	Number of examples	Top-N Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.502	0.30	0.238	26.26%
-e eki.txt	1176	0.516	0.271	0.236	14.92%
1. tekil kişi iyelik.txt	1128	0.640	0.221	0.148	49.60%
1. çoğul kişi iyelik.txt	1128	0.505	0.343	0.25	37.44%
2. tekil kişi iyelik.txt	1176	0.568	0.317	0.321	-1.13%
2. çoğul kişi iyelik.txt	1081	0.555	0.299	0.196	52.75%
3. tekil kişi iyelik eki.txt	1176	0.321	0.437	0.448	-2.37%
3. çoğul kişi iyelik.txt	1128	0.484	0.228	0.178	28.40%
eşitlik eki (-ce).txt	276	0.945	0.0345	0.027	27.84%
ismin -de hali.txt	1128	0.709	0.137	0.136	0.83%
ismin -i hali.txt	1176	0.337	0.439	0.439	0.18%
tamlayan eki.txt	1176	0.275	0.539	0.488	10.61%
vasıta eki (-le).txt	1128	0.454	0.337	0.253	33.32%
Overall	14005	0.510	0.318	0.274	16.11%

Table 7.5: *İsim çekim ekleri*: MRR Result Comparison for **Word2Vec** with 5 epochs

Morphological Categories	Number of examples	Top-N Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
Tamlayan eki	1176	0.252	0.548	0.488	12.30%
3. tekil kişi iyelik eki	1176	0.292	0.467	0.448	4.24%
İsmin -i hali	1176	0.325	0.454	0.439	3.42%
2. tekil kişi iyelik eki	1176	0.552	0.322	0.321	0.31%
Vasıta eki (-le)	1128	0.437	0.355	0.253	40.16%
1. çoğul kişi iyelik eki	1128	0.498	0.358	0.25	43.2%
-den eki	1128	0.481	0.316	0.238	32.77%
-e eki	1176	0.480	0.293	0.236	23.98%
2. çoğul kişi iyelik eki	1081	0.546	0.288	0.196	46.88%
3. çoğul kişi iyelik eki	1128	0.466	0.245	0.178	37.64%
1. tekil kişi iyelik eki	1128	0.654	0.216	0.148	45.95%
İsmin -de hali	1128	0.683	0.148	0.136	8.70%
Eşitlik eki (-ce)	276	0.956	0.028	0.027	3.70%
Overall	14005	0.479	0.329	0.273	20.51%

Table 7.6: *İsim çekim ekleri*: MRR Result Comparison for **Word2Vec** with 10 epochs

Semantic Categories	Number of examples	Top-N Miss Ratio	MRR
Aile	90	0.256	0.571
Para birimi	156	0.724	0.181
Şehir-bölge	1344	0.083	0.622
Ülke-başkent	506	0.045	0.683
Eş anlamlılar	600	0.385	0.476
Zıt anlamlılar	600	0.323	0.452
Overall	3296	0.210	0.551

Table 7.7: *Semantic categories*: MRR Results of **Word2Vec** with 10 epochs

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 82.53	p-value: 8.56×10^{-20}
	Spearman Result: 81.68	p-value: 4.11×10^{-19}
	OOV Ratio: 46.42	
Semantic Similarity	Pearson Result: 70.77	p-value: 4.77×10^{-57}
	Spearman Result: 74.57	p-value: 2.36×10^{-66}
	OOV Ratio: 26.6	

Table 7.8: Similarity test results for **Word2Vec** with 10 epochs ($p \times 100$)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.797	0.105	0.339	-69.03%
2. tekil kişi çekimi.txt	946	0.877	0.059	0.365	-83.84%
2. çoğul kişi çekimi.txt	946	0.851	0.085	0.327	-74.01%
3. çoğul kişi çekimi.txt	1128	0.803	0.105	0.42	-75.0%
emir kipi.txt	1176	0.656	0.21	0.465	-54.84%
gelecek zaman kipi.txt	1176	0.39	0.437	0.707	-38.19%
gereklilik kipi.txt	1128	0.632	0.212	0.467	-54.6%
geçmiş zaman eki (-di).txt	1176	0.4	0.428	0.535	-20.0%
geçmiş zaman eki (-miş).txt	1176	0.332	0.483	0.774	-37.6%
istek kipi.txt	1176	0.577	0.235	0.534	-55.99%
olumsuzluk eki.txt	1176	0.578	0.248	0.67	-62.99%
şimdiki zaman eki (-mekte).txt	1176	0.431	0.328	0.646	-49.23%
şimdiki zaman eki (-yor).txt	861	0.084	0.806	0.847	-4.84%
Overall	14231	0.567	0.287	0.549	-47.81%

Table 7.9: *Fiil çekim ekleri*: MRR Result Comparison for **left-aligned Word2Vec** with 10 epochs

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.727	0.148	0.238	-37.82%
-e eki.txt	1176	0.708	0.143	0.236	-39.41%
1. tekil kişi iyelik.txt	1128	0.768	0.123	0.148	-16.89%
1. çoğul kişi iyelik.txt	1128	0.596	0.254	0.25	1.6%
2. tekil kişi iyelik.txt	1176	0.634	0.229	0.321	-28.66%
2. çoğul kişi iyelik.txt	1081	0.693	0.182	0.196	-7.14%
3. tekil kişi iyelik eki.txt	1176	0.268	0.507	0.448	13.17%
3. çoğul kişi iyelik.txt	1128	0.549	0.196	0.178	10.11%
eşitlik eki (-ce).txt	276	0.967	0.018	0.027	-33.33%
ismin -de hali.txt	1128	0.832	0.073	0.136	-46.32%
ismin -i hali.txt	1176	0.341	0.431	0.439	-1.82%
tamlayan eki.txt	1176	0.352	0.411	0.488	-15.78%
vasıta eki (-le).txt	1128	0.679	0.164	0.253	-35.18%
Overall	14005	0.6	0.236	0.275	-14.09%

Table 7.10: *İsim çekim ekleri*: MRR Result Comparison for **left-aligned Word2Vec** with 10 epochs

Morphological Categories	Number of examples	Topn Miss Ratio	MRR
aile.txt	90	0.311	0.544
es-anlamlilar.txt	600	0.417	0.408
para-birimi.txt	156	0.788	0.157
sehir-bolge.txt	1344	0.085	0.617
zit-anlamlilar.txt	600	0.363	0.432
ülke-başkent.txt	506	0.067	0.703
Overall	3296	0.233	0.535

Table 7.11: *Semantic categories*: MRR Results of **left-aligned Word2Vec** with 10 epochs

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 79.05	p-value: 3.31×10^{-17}
	Spearman Result: 79.42	p-value: 1.85×10^{-17}
	OOV Ratio: 46.42	
Semantic Similarity	Pearson Result: 69.85	p-value: 5.20×10^{-55}
	Spearman Result: 73.98	p-value: 8.60×10^{-65}
	OOV Ratio: 26.6	

Table 7.12: Similarity test results for **left-aligned Word2Vec** with 10 epochs ($p \times 100$)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.323	0.418	0.339	23.3%
2. tekil kişi çekimi.txt	946	0.426	0.31	0.365	-15.07%
2. çoğul kişi çekimi.txt	946	0.316	0.41	0.327	25.38%
3. çoğul kişi çekimi.txt	1128	0.394	0.412	0.42	-1.9%
emir kipi.txt	1176	0.183	0.634	0.465	36.34%
gelecek zaman kipi.txt	1176	0.116	0.733	0.707	3.68%
gereklilik kipi.txt	1128	0.227	0.572	0.467	22.48%
geçmiş zaman eki (-di).txt	1176	0.242	0.586	0.535	9.53%
geçmiş zaman eki (-miş).txt	1176	0.117	0.677	0.774	-12.53%
istek kipi.txt	1176	0.319	0.477	0.534	-10.67%
olumsuzluk eki.txt	1176	0.122	0.668	0.67	-0.3%
şimdiki zaman eki (-mekte).txt	1176	0.334	0.391	0.646	-39.47%
şimdiki zaman eki (-yor).txt	861	0.072	0.863	0.847	1.89%
Overall	14231	0.244	0.551	0.549	0.43%

Table 7.13: *Fiil çekim ekleri*: MRR Result Comparison for **right-aligned Word2Vec** with 10 epochs

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.588	0.163	0.238	-31.51%
-e eki.txt	1176	0.361	0.317	0.236	34.32%
1. tekil kişi iyelik.txt	1128	0.738	0.154	0.148	4.05%
1. çoğul kişi iyelik.txt	1128	0.559	0.293	0.25	17.2%
2. tekil kişi iyelik.txt	1176	0.581	0.303	0.321	-5.61%
2. çoğul kişi iyelik.txt	1081	0.699	0.173	0.196	-11.73%
3. tekil kişi iyelik eki.txt	1176	0.298	0.394	0.448	-12.05%
3. çoğul kişi iyelik.txt	1128	0.579	0.148	0.178	-16.85%
eşitlik eki (-ce).txt	276	0.957	0.021	0.027	-22.22%
ismin -de hali.txt	1128	0.649	0.137	0.136	0.74%
ismin -i hali.txt	1176	0.262	0.421	0.439	-4.1%
tamlayan eki.txt	1176	0.259	0.513	0.488	5.12%
vasıta eki (-le).txt	1128	0.547	0.206	0.253	-18.58%
Overall	14005	0.515	0.266	0.275	-3.19%

Table 7.14: *İsim çekim ekleri*: MRR Result Comparison for **right-aligned Word2Vec** with 10 epochs

Morphological Categories	Number of examples	Topn Miss Ratio	MRR
aile.txt	90	0.367	0.454
es-anlamlilar.txt	600	0.492	0.348
para-birimi.txt	156	0.75	0.18
sehir-bolge.txt	1344	0.158	0.51
zit-anlamlilar.txt	600	0.39	0.457
ülke-başkent.txt	506	0.16	0.572
Overall	3296	0.295	0.463

Table 7.15: *Semantic categories*: MRR Results of **right-aligned Word2Vec** with 10 epochs

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 79.02	p-value: 3.50×10^{-17}
	Spearman Result: 76.72	p-value: 9.95×10^{-16}
	OOV Ratio: 46.42	
Semantic Similarity	Pearson Result: 70.11	p-value: 1.47×10^{-55}
	Spearman Result: 73.17	p-value: 9.61×10^{-63}
	OOV Ratio: 26.6	

Table 7.16: Similarity test results for **right-aligned Word2Vec** with 10 epochs ($p \times 100$)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.293	0.432	0.339	27.43%
2. tekil kişi çekimi.txt	946	0.452	0.298	0.365	-18.36%
2. çoğul kişi çekimi.txt	946	0.364	0.386	0.327	18.04%
3. çoğul kişi çekimi.txt	1128	0.375	0.394	0.42	-6.19%
emir kipi.txt	1176	0.193	0.627	0.465	34.84%
gelecek zaman kipi.txt	1176	0.105	0.781	0.707	10.47%
gereklilik kipi.txt	1128	0.247	0.561	0.467	20.13%
geçmiş zaman eki (-di).txt	1176	0.218	0.629	0.535	17.57%
geçmiş zaman eki (-miş).txt	1176	0.088	0.805	0.774	4.01%
istek kipi.txt	1176	0.22	0.599	0.534	12.17%
olumsuzluk eki.txt	1176	0.139	0.647	0.67	-3.43%
şimdiki zaman eki (-mekte).txt	1176	0.116	0.769	0.646	19.04%
şimdiki zaman eki (-yor).txt	861	0.064	0.889	0.847	4.96%
Overall	14231	0.217	0.606	0.549	10.44%

Table 7.17: *Fiil çekim ekleri*: MRR Results of **Word2Vec** with 15 epochs

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.473	0.335	0.238	40.76%
-e eki.txt	1176	0.458	0.307	0.236	30.08%
1. tekil kişi iyelik.txt	1128	0.614	0.24	0.148	62.16%
1. çoğul kişi iyelik.txt	1128	0.481	0.361	0.25	44.4%
2. tekil kişi iyelik.txt	1176	0.552	0.32	0.321	-0.31%
2. çoğul kişi iyelik.txt	1081	0.551	0.283	0.196	44.39%
3. tekil kişi iyelik eki.txt	1176	0.27	0.484	0.448	8.04%
3. çoğul kişi iyelik.txt	1128	0.477	0.235	0.178	32.02%
eşitlik eki (-ce).txt	276	0.949	0.031	0.027	14.81%
ismin -de hali.txt	1128	0.65	0.162	0.136	19.12%
ismin -i hali.txt	1176	0.313	0.464	0.439	5.69%
tamlayan eki.txt	1176	0.256	0.547	0.488	12.09%
vasıta eki (-le).txt	1128	0.427	0.361	0.253	42.69%
Overall	14005	0.468	0.337	0.275	22.67%

Table 7.18: *İsim çekim ekleri*: MRR Results of **Word2Vec** with 15 epochs

Morphological Categories	Number of examples	Topn Miss Ratio	MRR
aile.txt	90	0.244	0.59
es-anlamlilar.txt	600	0.382	0.468
para-birimi.txt	156	0.692	0.2
sehir-bolge.txt	1344	0.089	0.617
zit-anlamlilar.txt	600	0.332	0.458
ülke-başkent.txt	506	0.04	0.712
Overall	3296	0.212	0.555

Table 7.19: *Semantic categories*: MRR Results of **Word2Vec** with 15 epochs

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 81.54	p-value: 5.29×10^{-19}
	Spearman Result: 81.83	p-value: 3.14×10^{-19}
	OOV Ratio: 46.42	
Semantic Similarity	Pearson Result: 70.67	p-value: 5.33×10^{-57}
	Spearman Result: 74.53	p-value: 2.98×10^{-66}
	OOV Ratio: 26.6	

Table 7.20: Similarity test results for **Word2Vec** with 15 epochs

Morphological Categories	Number of examples	Top-N Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.087	0.675	0.339	99.16%
2. tekil kişi çekimi.txt	946	0.086	0.718	0.365	96.66%
2. çoğul kişi çekimi.txt	946	0.076	0.763	0.327	133.39%
3. çoğul kişi çekimi.txt	1128	0.277	0.621	0.42	47.99%
emir kipi.txt	1176	0.219	0.622	0.465	33.81%
gelecek zaman kipi.txt	1176	0.034	0.883	0.707	24.89%
gereklilik kipi.txt	1128	0.192	0.640	0.467	37.12%
geçmiş zaman eki (-di).txt	1176	0.256	0.589	0.535	10.16%
geçmiş zaman eki (-miş).txt	1176	0.080	0.759	0.774	-1.83%
istek kipi.txt	1176	0.161	0.559	0.534	4.76%
olumsuzluk eki.txt	1176	0.108	0.776	0.67	15.86%
şimdiki zaman eki (-mekte).txt	1176	0.058	0.884	0.646	36.91%
şimdiki zaman eki (-yor).txt	861	0.059	0.883	0.847	4.33%
Overall	14231	0.133	0.718	0.549	30.78%

Table 7.21: *Fiil çekim ekleri*: MRR Result Comparison for **FastText** with 5 epochs

Morphological Categories	Number of examples	Top-N Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.453	0.331	0.238	39.11%
-e eki.txt	1176	0.553	0.251	0.236	6.60%
1. tekil kişi iyelik.txt	1128	0.432	0.349	0.148	136.31%
1. çoğul kişi iyelik.txt	1128	0.295	0.512	0.25	105.19%
2. tekil kişi iyelik.txt	1176	0.482	0.339	0.321	5.91%
2. çoğul kişi iyelik.txt	1081	0.218	0.594	0.196	203.54%
3. tekil kişi iyelik eki.txt	1176	0.468	0.312	0.448	-30.20%
3. çoğul kişi iyelik.txt	1128	0.483	0.257	0.178	44.40%
eşitlik eki (-ce).txt	276	0.822	0.0747	0.027	176.77%
ismin -de hali.txt	1128	0.605	0.188	0.136	38.93%
ismin -i hali.txt	1176	0.458	0.332	0.439	-24.28%
tamlayan eki.txt	1176	0.276	0.515	0.488	5.56%
vasıta eki (-le).txt	1128	0.356	0.425	0.253	68.20%
Overall	14005	0.432	0.360	0.27	33.33%

Table 7.22: *İsim çekim ekleri*: MRR Result Comparison for **FastText** with 5 epochs

Semantic Categories	Number of examples	Top-N Miss Ratio	MRR
Aile	90	0.422	0.312
Para birimi	156	0.852	0.055
Şehir-bölge	1344	0.645	0.186
Ülke-başkent	506	0.193	0.548
Eş anlamlılar	600	0.611	0.243
Zıt anlamlılar	600	0.575	0.233
Overall	3296	0.560	0.258

Table 7.23: *Semantic categories*: MRR Results of **FastText** with 5 epochs

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 61.12	p-value: 5.70×10^{-9}
	Spearman Result: 62.55	p-value: 1.97×10^{-9}
	OOV Ratio: 46.42	
Semantic Similarity	Pearson Result: 67.45	p-value: 4.83×10^{-50}
	Spearman Result: 70.37	p-value: 3.82×10^{-56}
	OOV Ratio: 26.6	

Table 7.24: Similarity test results for **FastText** with 5 epochs ($p \times 100$)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.111	0.600	0.339	77.16%
2. tekil kişi çekimi.txt	946	0.096	0.671	0.365	83.91%
2. çoğul kişi çekimi.txt	946	0.104	0.713	0.327	118.16%
3. çoğul kişi çekimi.txt	1128	0.286	0.581	0.42	38.42%
emir kipi.txt	1176	0.222	0.588	0.465	26.62%
gelecek zaman kipi.txt	1176	0.036	0.871	0.707	23.26%
gereklilik kipi.txt	1128	0.189	0.647	0.467	38.58%
geçmiş zaman eki (-di).txt	1176	0.255	0.599	0.535	11.96%
geçmiş zaman eki (-miş).txt	1176	0.084	0.782	0.774	1.09%
istek kipi.txt	1176	0.182	0.534	0.534	0.11%
olumsuzluk eki.txt	1176	0.097	0.773	0.67	15.40%
şimdiki zaman eki (-mekte).txt	1176	0.063	0.850	0.64	31.65%
şimdiki zaman eki (-yor).txt	861	0.066	0.880	0.847	3.94%
Overall	14231	0.141	0.697	0.549	26.95%

Table 7.25: *Fiil çekim ekleri*: MRR Result Comparison for **FastText** with 10 epochs

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.443	0.332	0.238	39.78%
-e eki.txt	1176	0.520	0.282	0.236	19.53%
1. tekil kişi iyelik.txt	1128	0.469	0.321	0.148	117.29%
1. çoğul kişi iyelik.txt	1128	0.300	0.487	0.25	94.96%
2. tekil kişi iyelik.txt	1176	0.466	0.371	0.321	15.85%
2. çoğul kişi iyelik.txt	1081	0.248	0.561	0.196	186.27%
3. tekil kişi iyelik eki.txt	1176	0.393	0.372	0.448	-16.80%
3. çoğul kişi iyelik.txt	1128	0.469	0.259	0.178	45.95%
eşitlik eki (-ce).txt	276	0.884	0.054	0.027	103.33%
ismin -de hali.txt	1128	0.607	0.204	0.136	50.0%
ismin -i hali.txt	1176	0.406	0.382	0.439	-12.82%
tamlayan eki.txt	1176	0.256	0.549	0.488	12.62%
vasıta eki (-le).txt	1128	0.356	0.418	0.253	65.57%
Overall	14005	0.421	0.371	0.274	35.40%

Table 7.26: *İsim çekim ekleri*: MRR Result Comparison for **FastText** with 10 epochs

Semantic Categories	Number of examples	Top-N Miss Ratio	MRR
Aile	90	0.333	0.412
Para birimi	156	0.827	0.079
Şehir-bölge	1344	0.553	0.232
Ülke-başkent	506	0.138	0.583
Eş anlamlılar	600	0.573	0.276
Zıt anlamlılar	600	0.52	0.296
Overall	3296	0.493	0.303

Table 7.27: *Semantic categories*: MRR Results of **FastText** with 10 epochs

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 66.51	p-value: 7.58×10^{-11}
	Spearman Result: 68.16	p-value: 1.67×10^{-11}
	OOV Ratio: 46.42	
Semantic Similarity	Pearson Result: 67.35	p-value: 7.74×10^{-50}
	Spearman Result: 70.53	p-value: 1.71×10^{-56}
	OOV Ratio: 26.6	

Table 7.28: Similarity test results for **FastText** with 10 epochs ($p \times 100$)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.031	0.829	0.339	144.54%
2. tekil kişi çekimi.txt	946	0.045	0.787	0.365	115.62%
2. çoğul kişi çekimi.txt	946	0.056	0.793	0.327	142.51%
3. çoğul kişi çekimi.txt	1128	0.309	0.587	0.42	39.76%
emir kipi.txt	1176	0.165	0.701	0.465	50.75%
gelecek zaman kipi.txt	1176	0.017	0.921	0.707	30.27%
gereklilik kipi.txt	1128	0.117	0.754	0.467	61.46%
geçmiş zaman eki (-di).txt	1176	0.077	0.778	0.535	45.42%
geçmiş zaman eki (-miş).txt	1176	0.076	0.856	0.774	10.59%
istek kipi.txt	1176	0.104	0.712	0.534	33.33%
olumsuzluk eki.txt	1176	0.032	0.885	0.67	32.09%
şimdiki zaman eki (-mekte).txt	1176	0.052	0.9	0.646	39.32%
şimdiki zaman eki (-yor).txt	861	0.039	0.928	0.847	9.56%
Overall	14231	0.088	0.801	0.549	45.81%

Table 7.29: *Fiil çekim ekleri*: MRR Results of **FastText** word embeddings prepared by Facebook AI

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.302	0.476	0.238	100.0%
-e eki.txt	1176	0.336	0.464	0.236	96.61%
1. tekil kişi iyelik.txt	1128	0.35	0.462	0.148	212.16%
1. çoğul kişi iyelik.txt	1128	0.254	0.547	0.25	118.8%
2. tekil kişi iyelik.txt	1176	0.392	0.464	0.321	44.55%
2. çoğul kişi iyelik.txt	1081	0.188	0.65	0.196	231.63%
3. tekil kişi iyelik eki.txt	1176	0.25	0.562	0.448	25.45%
3. çoğul kişi iyelik.txt	1128	0.24	0.553	0.178	210.67%
eşitlik eki (-ce).txt	276	0.873	0.061	0.027	125.93%
ismin -de hali.txt	1128	0.374	0.333	0.136	144.85%
ismin -i hali.txt	1176	0.283	0.512	0.439	16.63%
tamlayan eki.txt	1176	0.15	0.726	0.488	48.77%
vasıta eki (-le).txt	1128	0.277	0.534	0.253	111.07%
Overall	14005	0.295	0.514	0.275	87.21%

Table 7.30: *İsim çekim ekleri*: MRR Results of **FastText** word embeddings prepared by Facebook AI

Morphological Categories	Number of examples	Topn Miss Ratio	MRR
aile.txt	90	0.211	0.564
es-anlamlilar.txt	600	0.392	0.398
para-birimi.txt	156	0.801	0.094
sehir-bolge.txt	1344	0.365	0.339
zit-anlamlilar.txt	600	0.428	0.4
ülke-başkent.txt	506	0.316	0.277
Overall	3296	0.39	0.346

Table 7.31: *Semantic categories*: MRR Results of **FastText** word embeddings prepared by Facebook AI

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 76.92	p-value: 4.64×10^{-15}
	Spearman Result: 80.28	p-value: 3.76×10^{-17}
	OOV Ratio: 49.28	
Semantic Similarity	Pearson Result: 65.12	p-value: 1.72×10^{-43}
	Spearman Result: 68.92	p-value: 1.72×10^{-50}
	OOV Ratio: 30.2	

Table 7.32: Similarity test results for **FastText** word embeddings prepared by Facebook AI

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.193	0.486	0.339	43.36%
2. tekil kişi çekimi.txt	946	0.184	0.525	0.365	43.84%
2. çoğul kişi çekimi.txt	946	0.172	0.555	0.327	69.72%
3. çoğul kişi çekimi.txt	1128	0.294	0.51	0.42	21.43%
emir kipi.txt	1176	0.194	0.631	0.465	35.7%
gelecek zaman kipi.txt	1176	0.078	0.832	0.707	17.68%
gereklilik kipi.txt	1128	0.209	0.63	0.467	34.9%
geçmiş zaman eki (-di).txt	1176	0.231	0.618	0.535	15.51%
geçmiş zaman eki (-miş).txt	1176	0.104	0.791	0.774	2.2%
istek kipi.txt	1176	0.204	0.577	0.534	8.05%
olumsuzluk eki.txt	1176	0.111	0.679	0.67	1.34%
şimdiki zaman eki (-mekte).txt	1176	0.08	0.807	0.646	24.92%
şimdiki zaman eki (-yor).txt	861	0.042	0.892	0.847	5.31%
Overall	14231	0.162	0.658	0.549	19.8%

Table 7.33: *Fiil çekim ekleri*: MRR Results of word vectors generated by averaging **Word2Vec** and **FastText** word vectors (10 epochs)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.45	0.33	0.238	38.66%
-e eki.txt	1176	0.487	0.299	0.236	26.69%
1. tekil kişi iyelik.txt	1128	0.534	0.287	0.148	93.92%
1. çoğul kişi iyelik.txt	1128	0.404	0.427	0.25	70.8%
2. tekil kişi iyelik.txt	1176	0.499	0.341	0.321	6.23%
2. çoğul kişi iyelik.txt	1081	0.396	0.425	0.196	116.84%
3. tekil kişi iyelik eki.txt	1176	0.335	0.448	0.448	0.0%
3. çoğul kişi iyelik.txt	1128	0.475	0.247	0.178	38.76%
eşitlik eki (-ce).txt	276	0.942	0.037	0.027	37.04%
ismin -de hali.txt	1128	0.637	0.177	0.136	30.15%
ismin -i hali.txt	1176	0.336	0.451	0.439	2.73%
tamlayan eki.txt	1176	0.246	0.548	0.488	12.3%
vasıta eki (-le).txt	1128	0.405	0.386	0.253	52.57%
çoğul eki.txt	1128	0.322	0.496	1.0	-50.4%
Overall	15133	0.434	0.368	0.329	12.02%

Table 7.34: *İsim çekim ekleri*: MRR Results of word vectors generated by averaging **Word2Vec** and **FastText** word vectors (10 epochs)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR
aile.txt	90	0.222	0.597
es-anlamlilar.txt	600	0.44	0.395
para-birimi.txt	156	0.731	0.144
sehir-bolge.txt	1344	0.196	0.508
zit-anlamlilar.txt	600	0.402	0.396
ülke-başkent.txt	506	0.113	0.625
Overall	3296	0.291	0.470

Table 7.35: *Semantic categories*: MRR Results of word vectors generated by averaging **Word2Vec** and **FastText** word vectors (10 epochs)

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 79.12	p-value: 1.98×10^{-51}
	Spearman Result: 79.19	p-value: 7.97×10^{-60}
	OOV Ratio: 46.43	
Semantic Similarity	Pearson Result: 68.15	p-value: 2.97×10^{-17}
	Spearman Result: 71.98	p-value: 2.67×10^{-17}
	OOV Ratio: 26.60	

Table 7.36: Similarity test results for word vectors generated by averaging **Word2Vec** and **FastText** word vectors (10 epochs)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.851	0.061	0.339	-82.01%
2. tekil kişi çekimi.txt	946	0.93	0.033	0.365	-90.96%
2. çoğul kişi çekimi.txt	946	0.925	0.028	0.327	-91.44%
3. çoğul kişi çekimi.txt	1128	0.717	0.117	0.42	-72.14%
emir kipi.txt	1176	0.563	0.266	0.465	-42.8%
gelecek zaman kipi.txt	1176	0.237	0.606	0.707	-14.29%
gereklilik kipi.txt	1128	0.673	0.158	0.467	-66.17%
geçmiş zaman eki (-di).txt	1176	0.273	0.48	0.535	-10.28%
geçmiş zaman eki (-miş).txt	1176	0.204	0.596	0.774	-23.0%
istek kipi.txt	1176	0.622	0.214	0.534	-59.93%
olumsuzluk eki.txt	1176	0.27	0.509	0.67	-24.03%
şimdiki zaman eki (-mekte).txt	1176	0.407	0.408	0.646	-36.84%
şimdiki zaman eki (-yor).txt	861	0.108	0.753	0.847	-11.1%
Overall	14231	0.512	0.33	0.549	-39.88%

Table 7.37: *Fiil çekim ekleri*: MRR Results of **GloVe** word vectors (100 iterations, window size 5)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.832	0.092	0.238	-61.34%
-e eki.txt	1176	0.746	0.117	0.236	-50.42%
1. tekil kişi iyelik.txt	1128	0.965	0.013	0.148	-91.22%
1. çoğul kişi iyelik.txt	1128	0.907	0.038	0.25	-84.8%
2. tekil kişi iyelik.txt	1176	0.721	0.166	0.321	-48.29%
2. çoğul kişi iyelik.txt	1081	0.99	0.003	0.196	-98.47%
3. tekil kişi iyelik eki.txt	1176	0.409	0.319	0.448	-28.79%
3. çoğul kişi iyelik.txt	1128	0.732	0.097	0.178	-45.51%
eşitlik eki (-ce).txt	276	0.989	0.006	0.027	-77.78%
ismin -de hali.txt	1128	0.856	0.057	0.136	-58.09%
ismin -i hali.txt	1176	0.434	0.317	0.439	-27.79%
tamlayan eki.txt	1176	0.466	0.333	0.488	-31.76%
vasıta eki (-le).txt	1128	0.881	0.062	0.253	-75.49%
çoğul eki.txt	1128	0.538	0.266	1.0	-73.4%
Overall	15133	0.73	0.144	0.329	-56.15%

Table 7.38: *İsim çekim ekleri*: MRR Results of **GloVe** word vectors (100 iterations, window size 5)

Semantic Categories	Number of examples	Top-N Miss Ratio	MRR
aile.txt	90	0.356	0.423
es-anlamlilar.txt	600	0.522	0.271
para-birimi.txt	156	0.923	0.037
sehir-bolge.txt	1344	0.141	0.53
zit-anlamlilar.txt	600	0.368	0.385
ülke-başkent.txt	506	0.081	0.722
Overall	3296	0.285	0.46

Table 7.39: *Semantic categories*: MRR Results of **GloVe** word vectors (100 iterations, window size 5)

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 73.45	p-value: 0.00 x 10
	Spearman Result: 74.13	p-value: 0.00 x 10
	OOV Ratio: 39.29	
Semantic Similarity	Pearson Result: 61.59	p-value: 0.00 x 10
	Spearman Result: 64.33	p-value: 0.00 x 10
	OOV Ratio: 23.60	

Table 7.40: Similarity test results for **GloVe** word vectors (100 iterations, window size 5)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.787	0.092	0.339	-72.86%
2. tekil kişi çekimi.txt	946	0.875	0.051	0.365	-86.03%
2. çoğul kişi çekimi.txt	946	0.86	0.058	0.327	-82.26%
3. çoğul kişi çekimi.txt	1128	0.658	0.157	0.42	-62.62%
emir kipi.txt	1176	0.529	0.309	0.465	-33.55%
gelecek zaman kipi.txt	1176	0.213	0.634	0.707	-10.33%
gereklilik kipi.txt	1128	0.598	0.222	0.467	-52.46%
geçmiş zaman eki (-di).txt	1176	0.297	0.467	0.535	-12.71%
geçmiş zaman eki (-miş).txt	1176	0.19	0.611	0.774	-21.06%
istek kipi.txt	1176	0.591	0.251	0.534	-53.0%
olumsuzluk eki.txt	1176	0.209	0.566	0.67	-15.52%
şimdiki zaman eki (-mekte).txt	1176	0.349	0.489	0.646	-24.3%
şimdiki zaman eki (-yor).txt	861	0.093	0.73	0.847	-13.81%
Overall	14231	0.472	0.363	0.549	-33.92%

Table 7.41: *Fiil çekim ekleri*: MRR Results of **GloVe** word vectors (100 iterations, window size 10)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.782	0.112	0.238	-52.94%
-e eki.txt	1176	0.71	0.123	0.236	-47.88%
1. tekil kişi iyelik.txt	1128	0.95	0.022	0.148	-85.14%
1. çoğul kişi iyelik.txt	1128	0.862	0.069	0.25	-72.4%
2. tekil kişi iyelik.txt	1176	0.697	0.185	0.321	-42.37%
2. çoğul kişi iyelik.txt	1081	0.975	0.01	0.196	-94.9%
3. tekil kişi iyelik eki.txt	1176	0.367	0.388	0.448	-13.39%
3. çoğul kişi iyelik.txt	1128	0.713	0.12	0.178	-32.58%
eşitlik eki (-ce).txt	276	0.989	0.008	0.027	-70.37%
ismin -de hali.txt	1128	0.833	0.065	0.136	-52.21%
ismin -i hali.txt	1176	0.386	0.384	0.439	-12.53%
tamlayan eki.txt	1176	0.425	0.361	0.488	-26.02%
vasıta eki (-le).txt	1128	0.828	0.088	0.253	-65.22%
çoğul eki.txt	1128	0.506	0.301	1.0	-69.9%
Overall	15133	0.697	0.171	0.329	-48.07%

Table 7.42: *İsim çekim ekleri*: MRR Results of **GloVe** word vectors (100 iterations, window size 10)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR
aile.txt	90	0.344	0.49
es-anlamlilar.txt	600	0.528	0.278
para-birimi.txt	156	0.878	0.071
sehir-bolge.txt	1344	0.092	0.614
zit-anlamlilar.txt	600	0.367	0.404
ülke-başkent.txt	506	0.061	0.712
Overall	3296	0.261	0.501

Table 7.43: *Semantic categories*: MRR Results of **GloVe** word vectors (100 iterations, window size 10)

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 69.87	p-value: 0.00 x 10
	Spearman Result: 72.33	p-value: 0.00 x 10
	OOV Ratio: 47.14	
Semantic Similarity	Pearson Result: 67.93	p-value: 0.00 x 10
	Spearman Result: 70.89	p-value: 0.00 x 10
	OOV Ratio: 26.60	

Table 7.44: Similarity test results for **GloVe** word vectors (100 iterations, window size 10)

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.666	0.103	0.339	-69.62%
2. tekil kişi çekimi.txt	946	0.573	0.125	0.365	-65.75%
2. çoğul kişi çekimi.txt	946	0.548	0.193	0.327	-40.98%
3. çoğul kişi çekimi.txt	1128	0.715	0.076	0.42	-81.9%
emir kipi.txt	1176	0.804	0.047	0.465	-89.89%
gelecek zaman kipi.txt	1176	0.405	0.329	0.707	-53.47%
gereklilik kipi.txt	1128	0.568	0.216	0.467	-53.75%
geçmiş zaman eki (-di).txt	1176	0.481	0.304	0.535	-43.18%
geçmiş zaman eki (-miş).txt	1176	0.444	0.374	0.774	-51.68%
istek kipi.txt	1176	0.542	0.198	0.534	-62.92%
olumsuzluk eki.txt	1176	0.545	0.211	0.67	-68.51%
şimdiki zaman eki (-mekte).txt	1176	0.279	0.51	0.646	-21.05%
şimdiki zaman eki (-yor).txt	861	0.365	0.373	0.847	-55.96%
Overall	14231	0.534	0.237	0.549	-56.82%

Table 7.45: *Fiil çekim ekleri*: MRR Results of static word vectors generated by **ELMo** trained on Turkish CoNLL17 corpus.

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.687	0.151	0.238	-36.55%
-e eki.txt	1176	0.635	0.23	0.236	-2.54%
1. tekil kişi iyelik.txt	1128	0.699	0.082	0.148	-44.59%
1. çoğul kişi iyelik.txt	1128	0.643	0.117	0.25	-53.2%
2. tekil kişi iyelik.txt	1176	0.665	0.14	0.321	-56.39%
2. çoğul kişi iyelik.txt	1081	0.461	0.077	0.196	-60.71%
3. tekil kişi iyelik eki.txt	1176	0.623	0.213	0.448	-52.46%
3. çoğul kişi iyelik.txt	1128	0.694	0.171	0.178	-3.93%
eşitlik eki (-ce).txt	276	0.511	0.009	0.027	-66.67%
ismin -de hali.txt	1128	0.666	0.116	0.136	-14.71%
ismin -i hali.txt	1176	0.677	0.178	0.439	-59.45%
tamlayan eki.txt	1176	0.498	0.3	0.488	-38.52%
vasıta eki (-le).txt	1128	0.612	0.191	0.253	-24.51%
çoğul eki.txt	1128	0.69	0.169	1.0	-83.1%
Overall	15133	0.633	0.162	0.329	-50.6%

Table 7.46: *İsim çekim ekleri*: MRR Results of static word vectors generated by **ELMo** trained on Turkish CoNLL17 corpus.

Morphological Categories	Number of examples	Topn Miss Ratio	MRR
aile.txt	90	0.856	0.066
es-anlamlilar.txt	600	0.928	0.029
para-birimi.txt	156	0.571	0.001
sehir-bolge.txt	1344	0.755	0.077
zit-anlamlilar.txt	600	0.657	0.214
ülke-başkent.txt	506	0.259	0.001
Overall	3296	0.687	0.078

Table 7.47: *Semantic categories*: MRR Results of static word vectors generated by **ELMo** trained on Turkish CoNLL17 corpus.

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 39.11	p-value: 2.00×10^{-3}
	Spearman Result: 39.43	p-value: 1.00×10^{-3}
	OOV Ratio: 55.00	
Semantic Similarity	Pearson Result: 35.18	p-value: 0.00 x 10
	Spearman Result: 36.77	p-value: 0.00 x 10
	OOV Ratio: 28.20	

Table 7.48: Similarity test results of static word vectors generated by **ELMo** trained on Turkish CoNLL17 corpus.

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
1. tekil kişi çekimi.txt	990	0.97	0.018	0.339	-94.69%
2. tekil kişi çekimi.txt	946	0.932	0.033	0.365	-90.96%
2. çoğul kişi çekimi.txt	946	0.928	0.04	0.327	-87.77%
3. çoğul kişi çekimi.txt	1128	0.961	0.019	0.42	-95.48%
emir kipi.txt	1176	0.959	0.016	0.465	-96.56%
gelecek zaman kipi.txt	1176	0.872	0.068	0.707	-90.38%
gereklilik kipi.txt	1128	0.946	0.026	0.467	-94.43%
geçmiş zaman eki (-di).txt	1176	0.966	0.02	0.535	-96.26%
geçmiş zaman eki (-miş).txt	1176	0.88	0.087	0.774	-88.76%
istek kipi.txt	1176	0.963	0.022	0.534	-95.88%
olumsuzluk eki.txt	1176	0.91	0.041	0.67	-93.88%
şimdiki zaman eki (-mekte).txt	1176	0.962	0.018	0.646	-97.21%
şimdiki zaman eki (-yor).txt	861	0.855	0.091	0.847	-89.26%
Overall	14231	0.932	0.038	0.549	-93.14%

Table 7.49: *Fiil çekim ekleri*: MRR Results of static word vectors generated by **Bert**.

Morphological Categories	Number of examples	Topn Miss Ratio	MRR	Reference MRR	Improvement with respect to reference
-den eki.txt	1128	0.938	0.027	0.238	-88.66%
-e eki.txt	1176	0.95	0.019	0.236	-91.95%
1. tekil kişi iyelik.txt	1128	0.968	0.014	0.148	-90.54%
1. çoğul kişi iyelik.txt	1128	0.965	0.013	0.25	-94.8%
2. tekil kişi iyelik.txt	1176	0.954	0.027	0.321	-91.59%
2. çoğul kişi iyelik.txt	1081	0.986	0.009	0.196	-95.41%
3. tekil kişi iyelik eki.txt	1176	0.895	0.059	0.448	-86.83%
3. çoğul kişi iyelik.txt	1128	0.938	0.027	0.178	-84.83%
eşitlik eki (-ce).txt	276	0.946	0.009	0.027	-66.67%
ismin -de hali.txt	1128	0.931	0.028	0.136	-79.41%
ismin -i hali.txt	1176	0.883	0.078	0.439	-82.23%
tamlayan eki.txt	1176	0.937	0.038	0.488	-92.21%
vasıta eki (-le).txt	1128	0.98	0.009	0.253	-96.44%
çoğul eki.txt	1128	0.812	0.123	1.0	-87.7%
Overall	15133	0.934	0.036	0.329	-89.07%

Table 7.50: *İsim çekim ekleri*: MRR Results of static word vectors generated by **Bert**.

Semantic Categories	Number of examples	Top-N Miss Ratio	MRR
aile.txt	90	0.767	0.137
es-anlamlilar.txt	600	0.848	0.072
para-birimi.txt	156	0.987	0.003
sehir-bolge.txt	1344	0.947	0.032
zit-anlamlilar.txt	600	0.968	0.017
ülke-başkent.txt	506	0.978	0.007
Overall	3296	0.935	0.034

Table 7.51: *Semantic categories*: MRR Results of static word vectors generated by **Bert**.

Similarity Task	Statistics	
Syntactic Similarity	Pearson Result: 25.17	p-value: 2.00 x 10^{-2}
	Spearman Result: 28.45	p-value: 8.00 x 10^{-3}
	OOV Ratio: 39.29	
Semantic Similarity	Pearson Result: 10.65	p-value: 3.70 x 10^{-2}
	Spearman Result: 12.63	p-value: 1.30 x 10^{-2}
	OOV Ratio: 23.60	

Table 7.52: Similarity test results of static word vectors generated by **Bert** trained on Turkish CoNLL17 corpus.

8. Conclusion

Based on our experimental results, we can come up with the following conclusions:

i. According to Bojanowski et. al., FastText architecture "ignores the internal structure of words, which is an important limitation for morphologically rich languages, such as Turkish or Finnish." [4] Because the FastText model is able to utilize character-level information, it excels at capturing the meaning of suffixes and prefixes, resulting in better performance for learning syntactic features such as noun and verb inflections. However, it is outperformed by Word2Vec models on semantic analogy tasks.

ii. In the original paper of GloVe [3], it is reported that GloVe outperforms Word2Vec. In our case however, we observed that GloVe embeddings were worse compared to word2vec. We found another study [18] with GloVe on a Turkish corpus which reported a similar finding.

It shows that, neural-network-based (predictive models) approaches are more suitable for training Turkish word embeddings compared to extracting the statistical information by training on the nonzero elements in a word-word cooccurrence matrix (count-based model).

iii. When all other metrics are held constant, there is no significant difference in performance between the CBOW and Skip-gram architectures of Word2Vec for Turkish verb conjugation suffixes. However, for noun declension suffixes, CBOW outperforms Skip-gram. On the other hand, for semantic analogy tasks, Skip-gram performs better than CBOW.

iv. Consistent with expectations, averaging the word vectors produced by Word2Vec and FastText with the same number of epochs resulted in word vectors that outperformed FastText in semantic analogy tasks and Word2Vec in syntactic analogy tasks.

v. Bommasani et al. (2020) [19] suggest two distinct methods for generating static word embeddings from contextual models. Similar to the results of their experiments for English, the **decontextualized** approach, in which each word w is considered independently of its context, performs poorly for Turkish as well.

References

1. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

2. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

3. Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

4. Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

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

6. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

7. Radim Rehurek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

8. Onur Güngör and Eray Yıldız. Linguistic features in turkish word representations. In *2017 25th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, 2017.

9. Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 271–276, Gothenburg, Sweden, May 2017. Association for Computational Linguistics.

10. Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium, October 2018. Association for Computational Linguistics.

11. Frédéric Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 246–252, 2005.

12. Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. volume 41, pages 391–407.

13. Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

14. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

15. H. Sak, Andrew Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 338–342, 01 2014.

16. Aylin Üstün, Murat Kurfalı, and Burcu Can. Characters or morphemes: How to represent words? In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 144–153, 2018.

17. Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

18. Murat Aydogan and Ali Karci. Kelime temsil yöntemleri ile kelime benzerliklerinin İncelenmesi. *Çukurova Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 34:181–195, 06 2019.

19. Rishi Bommasani, Kelly Davis, and Claire Cardie. Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4758–4781, Online, July

2020. Association for Computational Linguistics.