ENRICHING CONTEXTUAL WORD EMBEDDINGS WITH CHARACTER INFORMATION

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Ozan POLATBİLEK

July 2020
İZMİR

To my beloved wife

ACKNOWLEDGMENTS

Some part of this thesis is done on TITAN V GPU which is granted by Nvidia Research Grant Program.

The TÜBİTAK 117E747 Yürüme Verisi ve Konum Mahremiyeti Project helped me to understand fundamentals of temporal data.

I, personally, thank Erhan Sezerer for technical support in this thesis and my Advisor for her endless support.

ABSTRACT

ENRICHING CONTEXTUAL WORD EMBEDDINGS WITH CHARACTER INFORMATION

Natural Language Processing has become more and more popular with the recent advances in Artificial Intelligence. Fundamental improvements have been introduced in word representations to store semantic and/or syntactic features. With the recently published language model BERT, contextual word vectors could be generated. This model do not process character level information. In morphologically rich languages such as Turkish, this model's perception of syntax could be improved.

In this thesis, a new model, called BERT-ELMo, which is a combination of BERT and ELMo, is proposed to enrich BERT with character level information. This model combines character level processing part of ELMo and contextual word representation part of the BERT model. To show the effectiveness of the proposed model, both quantitative (question answering) and qualitative (word analogy, word contextualization, morphological meaning, out of vocabulary word capturing) analyses are performed and it is compared with BERT on Turkish language. Thanks to character level addition, proposed model is able get trained in any language without any pre-analysis.To the best of our knowledge, this is the first study which uses morphological analysis to train the BERT model in Turkish, and the first model to integrate a character level module to BERT.

ÖZET

BAĞLAMSAL KELİME GÖMMELERİNİN KARAKTER BİLGİSİ İLE ZENGİNLEŞTİRİLMESİ

Doğal dil işleme, günümüzdeki yapay zeka gelişmelerinin de yardımıyla popülerlik kazanmıştır. Bu popülerlik sayesinde farklı alt alanlarda yüksek başarımlı çalışmalar yayınlanmaktadır. Bu alt alanlardan birisi de doğal dil işlemenin temel problemlerinden birini ele alan kelimelerin vektörel gösterimi alanıdır. Kelimelerin sözdizimi ve anlamsal bilgilerini ihtiva etmesi amacıyla vektörel gösterim kullanılmaktadır. Yakın zamanda yayınlanan BERT modeli ile kelime vektörleri bağlamsal olarak yüksek bir başarım ile gösterilebilmektedir. Öte yandan bu model sözdizimi kurallarını gözlemleyecek karakter seviyesinde bir yapı içermediğinden, Türkçe gibi morfolojik olarak zengin dillerde istenen sonuçları veremeyebilmektedir.

Bu çalışmada, BERT modelinin karakter seviyesinde de bilgi işleyebilmesini sağlamak amacıyla, BERT-ELMo modeli önerilmiştir. Bu yeni önerilen modelde, ELMo modelinin karakter seviyesinde işlem yapabilen modülü ile BERT modelinin bağlamsal kelime vektörü üreten modülü birleştirilmiştir. BERT-ELMo modeli hem nicel and nitel analizler ile incelenmiş ve BERT modeli ile Türkçe dili üzerinden karşılaştırılmıştır. Karakter seviyesindeki ekleme sebebiyle, önerilen model herhangi bir dilde herhangi bir ön çalışma yapılmaksızın çalıştırılabilir. Bilinen kadarıyla, bu çalışma Türkçe morfolojik analizi ile BERT modelini eğiten ve BERT modeline karakter seviyesinde bir modül eklemeyi deneyen ilk çalışmadır.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
BERT	Bidirectional Encoder Representations from Transformers
ELMo	Embeddings from Language Models
ReLu	Rectified Linear Unit
OOV	Out of Vocabulary
FCNN	Fully-Connected Neural Network
NLP	Natural Language Processing
BiLSTM	Bidirectional Long Short-Term Memory
SOTA	State of the Art
GLUE	General Language Understanding Evaluation Benchmark
NSP	Next Sentence Prediction
SQuAD	Stanford Question Answering Dataset
LM	Language Modeling
NLU	Natural Language Understanding
AI	Artificial Intelligence
PCA	Principle Component Analysis

CHAPTER 1

INTRODUCTION

Since the beginning of computer science, creating a artificial intelligence (AI) has always been a desire. With the study of (Allen M. Turing, 1950), the basics of intelligence in machines are introduced and what to expect from a machine in terms of intelligence is discussed. After that paper, AI domain has gotten popularity in public. Several movies, books are published to talk about fictional stories of AI. The one common thing in those artistic products is creating a robot which is exactly (sometimes just closely) same as human in every terms, physical or mental. Then AI domain is divided into sub-parts since the question of "what makes human different from machine" has several different answers. In the way to reach a human-like machine, one should achieve multiple features of human that makes it intelligence, such as vision, perception, physical ability, consciousness, being able to speak etc.

During 1960's, processing human language, which is called as natural language processing (NLP), advanced since AI is progressed rapidly in those times. With the studies (Noam Chomsky, 1957), (John McCarthy, 1960); processing natural language with machines became a reality and scientists had gone one step towards making machine understand natural language. With these advancements, dividing sentences into words (tokenization), determining which part-of-speech (verb, noun, adjective etc.) a word stands for (POS tagging), dividing word into its root and suffixes (stemming and lemmatization) were automated. Although with these progresses machines were able to parse strings or divide sentences into its structural parts, they were not able to interpret the meaning of a word. A word is an atomic element of the speech that stands for a meaning on its own. This meaning might be concrete (originated from physical world) or from an abstraction. With the study of (Gabriella Vigliocco et al., 2009), it is shown that children firstly acquire concrete words, such as *food*, *water*, *car*, since they are easy-to-bind with what has seen. As they develop their language abilities, they start to acquire abstract words, like *institution*, *god*, *love*.

To define how the meaning of a word is constructed, (Zellig S. Harris, 1954) suggested distributional hypothesis which is popularized by the study of (John R. Firth, 1957). The hypothesis states that "the meaning of a word is defined by the words surrounding it". One analogy of this hypothesis would be thinking of describing a word without using some specific words. For example, describing what the word *coffee* is without using words: *caffeine*, *milk*, *tea*, *hot*, *glass*, *water*. This example shows us that it is very hard to define a word without using words that are frequently used with it, so the meaning of the target word has strong bindings with the words that are frequently used with it. By combining this hypothesis with statistical approaches, scientists came up with mathematical models to extract and store the meaning of words.

Mathematically, words are represented with a vector of features. The most basic representation type is an one-hot vector notation. In this notation, length of the vector is the length of the vocabulary, and all the units of vector is zero except the index that corresponds to the target word. For a three word vocabulary of {*computers*, *are*, *awesome*}, vector for *computers* would be [1, 0, 0], for *are* [0, 1, 0] and for *awesome* [0, 0, 1]. This representation is called as local representation, since the word is represented with a single node and that node is only linked to that word. However, (Geoffrey E. Hinton et al., 1986) study showed that a distributed representation where units are features and each object (or word) is represented with collection of feature activations could be more easy-to-compute and the vector would be rich in terms of information. In a distributed representation the vector would be denser and shorter with respect to the local representation. For example, to represent all animals in the nature, one can use features like *hairy*, *number of legs*, *mammal*, *carnivore*, *omnivore*. To represent a cat, vector of [0.6, 4, 1, 1, 0] can be used, where features placed respectively, or to represent a dolphin, vector of [0, 0, 1, 1, 0] can be used. As shown in the example, the vector consists of features and the elements in the domain (in our example element is animal and domain is nature) are representable in terms of those feature values.

One way of creating a distributed representation is count based methods. They are the fundamental methods and used commonly in early days of NLP. The idea of these methods comes from the distributional hypothesis: if the meaning of the words are constructed with their surrounding words, then one could count how many times other words

used with the target word. The most basic type of count based method is bag-of-words representation. In bag-of-words, first content words (word that holds for a meaning on their own) are determined, then by counting how many times those words occur, a document representation is created. The very same idea can be applicable for word representation if a window is used (a sentence, or might be N-word long window) instead of a document. So the word frequencies are determined in every window which has the target word in it. Various statistical operations can be done on the vector to make it more useful, such as tf, tf-idf, PCA etc.

Another method of count based approach is N-gram models. N-grams are word word groups where N represents the size of the group and this time frequency of these N-grams are determined. To determine frequencies, first N-gram groups are determined through iterating a window, length of N, over text. Then while iterating this window, the frequency of each encountered N-gram is gathered. This approach is more local context oriented than bag-of-words model which is global context (all text) oriented. For example, when we look at the sentences: "*science can set us free and break the chains of ignorance*", and "*ignorance can break the chains of science and set us free."*, they mean completely different to each other but their bag-of-words representations would be same in the scale of global context. By ordered-grouping words (N-grams), solving short dependencies are enforced, so we have phrases or word pairs.

Apart from count-based method previously discussed, there is another approach in AI and of course in NLP: Artificial Neural Networks. Artificial Neural Networks (or just neural networks in short) are an adaptation of how human brain works. Human brain has neuron and synapses between those neurons. Due to electrical signals that flows through one neuron to another, what we call thinking occurs. To mimic this brain activity, a network which is called as perceptron is suggested by (Frank Rosenblatt, 1957). A perceptron is an analogy of group of neurons (input signals in our case) weighted by a factor and turning into an output signal. Network learns how to adjust weights in a way that a certain input signal results in high output signal in its corresponding class. To adjust the weights, network receives some example data with their actual class information and the network tries to predict the true class value of data at each iteration. By learning from its success and failures, network advances trough iterations. This procedure is called as

training since the network trains itself to find the best-possible weight distribution. Then with the data which are not included training data it fed to the network and the results of the predictions are measured by a scoring function, and this procedure is called as testing. With training by training data and testing with the rest, a neural network model can be executed and recently neural models are quite successful in most of the fields of AI.

Processing a text with a perceptron given above is not effective since the idea or message inside a text is formed up after all the words and sentences in it. A perceptron, only evaluates current data without concerning about past or future data, however in natural language, there are dependencies of words to other words. In a phrase of "*The lord of the rings*", word *lord* has a relationship with the word *ring*, but they are three word away from each other. To process natural language, one needs to consider the past events (words) to predict the future words. (Jeffrey L. Elman, 1990) introduced recurrent neural networks (RNN) to process temporal events and showed that next bit prediction problem could be solvable by it. In its example, author conducts an RNN that receives bit signal sequence. After every bit, it makes a prediction, however the goal is to predict XOR output of past 2 bits for the model. An example of input-output sequence is shown below:

Input : 0 1 **1** 1 0 **1** 0 0 **0** 1 1 **0**

Output: 1 1 **1** 0 0 **1** 0 1 **0** 0 0 **0**

For the network probability of predicting next bit is 0.5 for first bit, but when second bit arrives then third bit became predictable since it must be the result of previous two bits. This way author proved that an RNN would solve temporal dependency which also occurs in NLP as well. So what author achieved is called as language model where model is expected to produce next word with given N words with respect to the language rules. Later, (Yoshua Bengio et al., 2003) adapted this problem into NLP and defined the probabilistic language modeling task where the goal is to predict N+1'th word with given N words.

By using this language modeling idea, (Tomas Mikolov et al., 2013) study proposed a neural network model, called as Word2Vec, to create a distributed representation. There are two versions of this model; CBOW and skip-gram. In CBOW, the task is to predict *N*th word by given the company of the *N*th word. To predict what the word at

[MASK] should be in the sentence of "Rabbits have hairy, [MASK], adorable ears.", the words *Rabbits*, *hairy*, *adorable*, *ears* are processed. In skip-gram model, it is vice versa. The target word is given to the model and the surrounding words are predicted. With this study, it is shown that neural networks are good at creating a word vector and they started to gain popularity in NLP.

Words get a meaning within a context. For example, word *bank* have two different meanings in following sentences: "He went to the bank to learn their interest rates" and "We sat on a bank until sun dawn". In the first sentence *bank* means the place where people deposit or withdraw money from it, but in the second it means a bench. To represent these two different meanings, two different vectors are needed. However, models like Word2Vec (Tomas Mikolov et al., 2013) or GloVe (Jeffrey Pennington et al., 2014) only create global vectors, one vector only for one word. To represent different senses of a word, scientist had mainly focused on solutions with WordNet (George A. Miller, 1995) until transfer learning of language models became popular.

Although the models mentioned so far only deals with semantic meaning of a word, words have another meaning type called as syntactic. Syntax is rules that the language applies to its elements like words, phrases or sentences. These syntactic rules stand for a meaning as well. A nice example would be thinking of suffix "*s*" in the English language. It makes the word plural when it is added at the end of the word. So examining whether *s* occurs at the end of the word provides an additional meaning. Human do not perceive words like *book* and *books* as separate elements, so to mimic this behavior, models should analyse syntactic meaning as well. In the study of (Kim Yoon et al., 2016), authors used Convolutional Neural Networks (CNN) (Yann Lecun and Yoshua Bengio, 1995) to analyse characters of the words and create a syntactic representation of a word. After that with a Highway Neural Network (Rupesh Kumar Srivastava et al., 2015), they managed to combine the syntactic representation from CNN and the semantic meaning representation from GloVe (GloVe is a model that creates global vectors like Word2Vec). As a result, they get the vector representation of a word which both consists of semantic and syntactic meaning.

In the golden year (2018) of NLP, a lot of different models and approaches are proposed about word representations.The paper of (Matthew Peters et al., 2018) proposed

a model named as ELMo which used LSTM (Sepp Hochreiter and Jürgen Schmidhuber, 1997) (an advanced type of RNN) to obtain contextual word representation. That means, for two different meaning of word *bank*, the model can provide 2 different word vectors as their meaning shift from sentence to sentence. This model is highly used with transfer learning where one uses learned features of a model in any other task with different model. In ELMo's case, the contextual word representation gathered from ELMo is used as input value to several different architectures and outperformed most of the state-of-the-art models on several Natural Language Understanding tasks in GLUE benchmark (Alex Wang et al., 2018) with high margin. Another improvement of ELMo was to create a model that perceives in character level but generates at word level. That means model processes characters of a word then creates a contextual representation with respect to the context. The advantage of this approach is to make the model understand suffixes or compound words, such as birthday, airplane, database, without any preprocessing.

In the very same year, BERT model (Jacob Devlin et al., 2018) is suggested and it uses transformer network (Ashish Vaswani et al., 2017) to create contextual word representation. The model relied on previously done morphological analysis to perceive character level features. In morphologically rich languages, like Turkish or Finnish, these morphological analyses have not done yet and they are an open problem for those languages as well. Even though, BERT model creates more successful representations, it has problem of processing out of vocabulary (OOV) words and complex words, like words with several suffixes.

This study has two main contributions. One is to overcome the problems of the BERT model in processing OOV words and complex words in morphologically rich languages, this study proposes a new hybrid model of ELMo and BERT. Proposed model uses character-level Convolutional Neural Network as ELMo does where characters of a word is processed and a word vector is produced. Then this word vector is passed to BERT model to get the contextual embedding of the word. The second one is using Turkish Language as a target language for contextual representation with transformer models for the first time (as far as we know) and making a morphological analysis on Turkish to make it suitable to use it with these kind of networks. Both qualitative; how well the vector space is constructed, how much vectors satisfy contextual meaning of the word,

and quantitative; how much score do they get in other tasks, analyses are conducted on Turkish to compare the morphological information gathering better.

1.1. Organization of Thesis

In Chapter 2, the neural network types that are used in thesis are explained in detail. The level of detail is kept in a level that an introduction to machine learning, algebra and vectoral operation knowledge would be enough to understand. After introducing background models, in Chapter 3 related works are explained in word embedding, contextual word embedding and character-word mixture language model fields. This chapter divided into two unrelated topics "character and word level combinations" and "contextual embeddings". Since proposed model includes both of these phenomena, they are investigated differently. After related work, Methodology is explained in Chapter 4. Methodology chapter explains the proposed model, its intuition, training procedures, evaluation procedures. Then in Chapter 5, the results of these evaluations are shown and what the results indicate is discussed. Finally, Chapter 6 makes a conclusion of what this study did and how the contributions should be interpreted.

CHAPTER 2

BACKGROUND MODELS

There are several different types of neural network architectures. Some of them are better at temporal data, some of them are good at local patterns. Although their usage varies with the problem they are applied on, their functionality and how they learn is well-defined. All the neural network architectures which are used in this thesis is explained in detail in this chapter.

2.1. Long Short-Term Memory (LSTM)

Recurrent Neural Networks (RNN) (Jeffrey L. Elman, 1990) are the neural network architectures that can encode temporal data. For every time step, RNN takes both input of that time step (x_t) and context vector (C_{t-1}), then outputs the next context vector (C_t) as a result. With this approach, when we get the last context vector output, we can assume that we encode all processed information in it. It is like vector representation of a text, or a vector representation of time series data. So RNN is a neural network to process temporal data.

Long Short-Term Memory (LSTM) (Sepp Hochreiter and Jürgen Schmidhuber, 1997) is a neural network architecture which is a successor of RNN. The study by (Sepp Hochreiter, 1998) showed that vanilla (the first version) RNNs have the problem of vanishing gradients. The vanishing gradient problem is a phenomenon that happens because of multiplication of gradients, which are small numbers between 1 and 0, and becomes even smaller as time moves forward. Since gradients are small numbers, when they are multiplied at each time step, it gets smaller and smaller until it has no effect, as a result of which learning stops. LSTM solves this problem with its forget, input and output gate formation. This structure in the form of a cell representation is shown in Figure 2.1.

In LSTM; forget gate (f gate) is responsible of how much information to use in

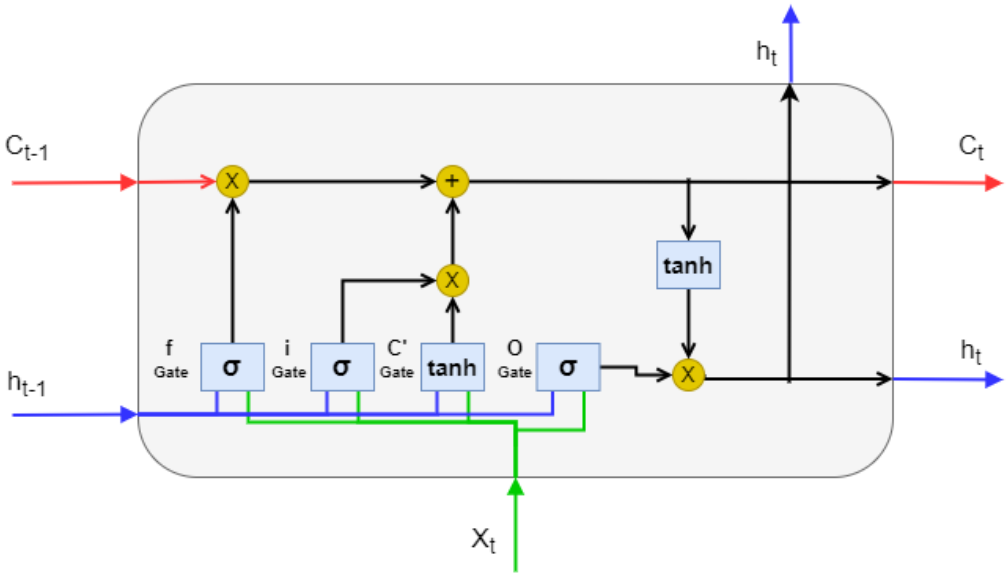

Figure 2.1. LSTM (Sepp Hochreiter and Jürgen Schmidhuber, 1997) Cell in detail.

this time step, input gate (i gate) controls which information (hidden (H_{t-1}) or input (X_t)) will go into the cell, cell gate (C gate) forms new cell output (C_t) with the help of output of f gate and i gate and finally output gate (o gate) creates the next hidden output (H_t). The equations for functions inside LSTM are given in Equation 2.1. In Eq. 2.1, σ and $tanh$ are nonlinear transformations where the former produces an output between 0 and 1 and the latter works with an output range between -1 and 1. Their formulae are given in Equation 2.2.

$$
\begin{aligned}
i_t &= \sigma(x_t U^i + h_{t-1} W^i) \\
f_t &= \sigma(x_t U^f + h_{t-1} W^f) \\
o_t &= \sigma(x_t U^o + h_{t-1} W^o) \\
\tilde{C}_t &= tanh(x_t U^g + h_{t-1} W^g) \\
C_t &= \sigma(f_t * \tilde{C}_{t-1} + i_t \tilde{C}_t) \\
h_t &= tanh(C_t) * o_t
\end{aligned}
\tag{2.1}
$$

$$\sigma(x) = \frac{e^x}{e^x + 1}$$
$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

LSTM cells are used to encode temporal data such that one cell's hidden output will be its successor's hidden input. By the processing of the input of that time step and previous hidden, new hidden vector output is created. As shown in Figure 2.2, this architecture processes inputs through time and each hidden vector that is directed upward is sent to the prediction phase. Prediction phase is generally a feed-forward neural network with no hidden layer (sometimes one), whose output size is the number of classes. After prediction, error of the network at time t (at time 1 in our figure) is calculated and backpropagated (Henry J. Kelley, 1960) through cell. Thus, the learning process can be summarized as follows: At each time step; hidden output is calculated in gates with the previous hidden and current input, then a prediction is created by a Fully-Connected Neural Network (FCNN) (Frank Rosenblatt, 1957) and finally loss function computes an error and that error is backpropagated to adjust the weights of the cells and FCNN. An example of loss function is given Equation 2.3 which is called cross-entropy loss. Cross-entropy loss is a function with two sides. Error from true label and error from wrong label. If $y = 1$ that means this is the true label and the prediction of the model should be as high as possible. Any value that is far from 1 will be punished in logaritmic manner. The right hand part of the summation will not produce error since $(1 - y)$ part would zero out the log error. When $y = 0$ label is incorrect, so only the right hand part will produce error. If $y = 0$ then model should produce as less probability as possible. So any probability other than 0 will be punished. The sum of errors in all labels creates the Loss of the model in that data point. This Los function forces model to predict 1 for true label and 0 for others. That makes is suitable for multi-class, single-label classification.

$$Loss(p|y) = -(ylog(p) + (1 - y)log(1 - p))$$
Where p = prediction vector, y = ground-truth vector (2.3)

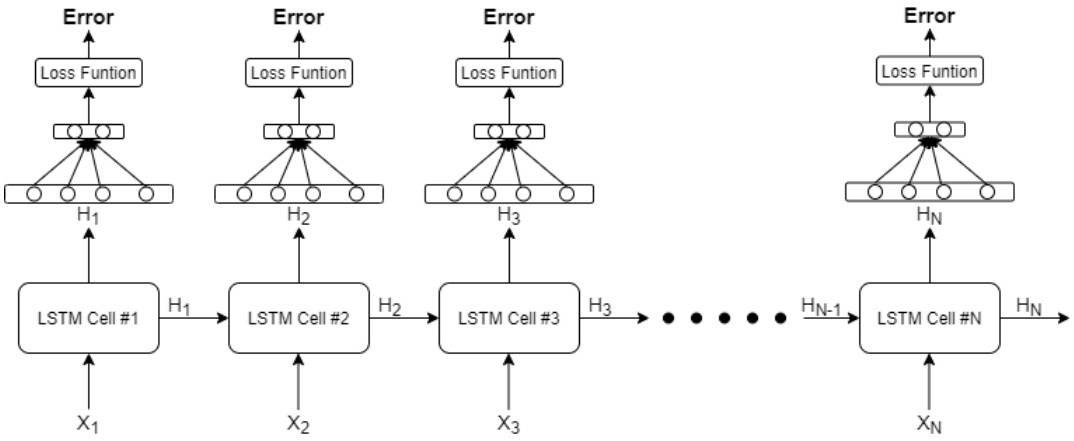

Figure 2.2. LSTM Sequence.

2.2. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) (Yann Lecun and Yoshua Bengio, 1995) is an architecture that captures local features in the data. This architecture gets its name from the nature of "weight matrices that convolve on data". As shown in Figure 2.3, process starts with weights matrices to scan the data. Scanning means the multiplication of weight matrix with the current local square part of the data. This scanning process starts from the origin (0,0) (for a 2-D convolution) point and processes all the possible squares of data. Each scan results in a corresponding point at feature map. There are N number of feature maps for N weight matrices. Working sizes for each step can be followed by the numbers below the layers in Figure 2.3.

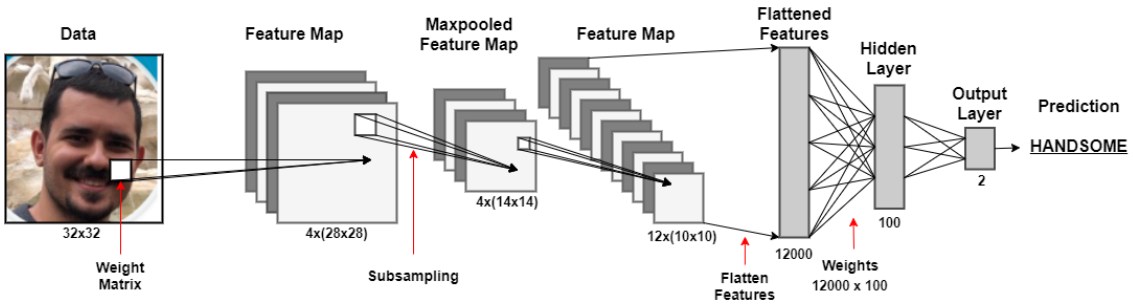

Figure 2.3. An Example of CNN (Yann Lecun and Yoshua Bengio, 1995) process.

After creating a feature map, a subsampling process called maxpooling is per-

formed. It subsamples the max value inside its scope. For example, if max-pooling is 2x2 in size and its scope is $[[2, 1], [0, 4]]$ at that time, then 4 is taken out and written to the new feature map. This process reduces the number of features by selecting highly activated signals since they will contribute much to the prediction. This convolution-maxpooling process goes like a chain with any number of layers and then in the final step, all the features in all feature maps are flattened into one vector. This vector is a representation of the input data. So this representation is fed into a FCNN to make a prediction.

CHAR \ FEATURES	#1	#2	#3	#4	#5	#6	#7
H	0.1	0.45	0.22	0.01	0.9	0.2	0.11
A	0.04	0.5	0.5	0.23	0.73	0.1	0.88
N	0.12	0.2	0.74	0.73	0.53	0.42	0.09
D	0.09	0.92	0.25	0.45	0.19	0.21	0.6
S	0.85	0.2	0.65	0.21	0.61	0.77	0.35
O	0.94	0.95	0.29	0.05	0.02	0.50	0.27
M	0.99	0.26	0.87	0.31	0.61	0.69	0.22
E	0.51	0.21	0.05	0.42	0.1	0.85	0.73

(a) First step

CHAR \ FEATURES	#1	#2	#3	#4	#5	#6	#7
H	0.1	0.45	0.22	0.01	0.9	0.2	0.11
A	0.04	0.5	0.5	0.23	0.73	0.1	0.88
N	0.12	0.2	0.74	0.73	0.53	0.42	0.09
D	0.09	0.92	0.25	0.45	0.19	0.21	0.6
S	0.85	0.2	0.65	0.21	0.61	0.77	0.35
O	0.94	0.95	0.29	0.05	0.02	0.50	0.27
M	0.99	0.26	0.87	0.31	0.61	0.69	0.22
E	0.51	0.21	0.05	0.42	0.1	0.85	0.73

(b) Second step

CHAR \ FEATURES	#1	#2	#3	#4	#5	#6	#7
H	0.1	0.45	0.22	0.01	0.9	0.2	0.11
A	0.04	0.5	0.5	0.23	0.73	0.1	0.88
N	0.12	0.2	0.74	0.73	0.53	0.42	0.09
D	0.09	0.92	0.25	0.45	0.19	0.21	0.6
S	0.85	0.2	0.65	0.21	0.61	0.77	0.35
O	0.94	0.95	0.29	0.05	0.02	0.50	0.27
M	0.99	0.26	0.87	0.31	0.61	0.69	0.22
E	0.51	0.21	0.05	0.42	0.1	0.85	0.73

(c) Third step

Figure 2.4. First three steps of Char CNN (Red borders are convolutions).

In NLP, character groups form morphemes and these morphemes can vary in length. Also, words can have multiple morphemes in it. So a system which decomposes words into its morphemes and learn those morphemes would be better to capture both

syntactic and semantic differentiation of words. Since CNNs are good at local pattern recognition, in NLP they are used to capture morpheme information. So a CNN looks at words character by character.

To learn a character CNN representation for a word, character embeddings are gathered and listed in their order in the word. Then a window (kernel) starts to scan the characters, however different from a vision task, the width of the window is equal to the length of embeddings. Because embeddings are meaningful when contains the features all together, dividing an embedding would result in creating different vector space. So this embedding-wide and pre-set height window does what CNN does. The first three steps of the respective processing are shown in Figure 2.4. After processing all the characters, one can concatenate or perform another operation on feature maps of CNN to get a vector representation of the word gathered from its characters.

2.3. Highway Neural Networks

In normal FCNN, at each layer a non-linear transformation is applied to the previous activations. Later, scientists found that residual blocks (passing just the activation to the next layer without performing anything on it), help to regulate the network and enable to build a deeper one. However, they are pre-determined structures so there is no intelligence, you might force the network to pass data although it has a critical activation for the features. So (Rupesh Kumar Srivastava et al., 2015) come up with an idea to control this pass-or-activate trade-off. They introduced highway networks where network learns an additional feature: how much information to pass or process. In mathematical terms:

If we define FCNN as below without biases to simplify:

$y = A(x, W_A)$

Then a highway network looks like the one below (where $\odot$ is pointwise matrix multiplication):

$y = A(x, W_A) \odot T(x, W_T) + x \odot (1 - T(x, W_T))$

This way network learns how much to transform (process) and pass the rest. This model is quite useful in the case of passing one sub-network's output to a higher one.

Here x is the highway network input, W_A is the normal weight matrix for the FCNN, T is the operation that determines how much data to be transformed (multiplied by the weights of FCNN). The right hand part of the addition is something like residual connection. For example, if T outputs as 0.75, this means 0.75 of the data will be transformed and 0.25 of it will be directly passed (like residual connection).

2.4. Attention Mechanism

Attention architecture (Dzmitry Bahdanau et al., 2014) is a neural network which highlights the most important features of a vector inside a neural network architecture. It is generally used in encoder-decoder networks since they consist of information bottlenecks (all the information created by an encoder is narrowed down into the final module, so it is a bottleneck). In these bottlenecks, attention mechanism is favorable to create more refined encodings. The architecture got popularity after the paper and several studies improved their models by just adding an attention in it. It also provides its highlights in the form of attention heat maps and these heat maps are used to add explainability to neural network models. Just looking at activations tells only how the features are constructed but the effect of certain features on the prediction can be shown with attention.

There are multiple attention types: Additive, dot-product, scaled dot-product, and multi-head. Since dot-product is quite similar to scaled dot-product, we will not explain it further. It is exactly the same as scaled dot-product without its scaling factor. Additive attention does highlighting by learning what to highlight in W_a matrix at each time step. The process is shown in Figure 2.5. As can be seen in Eq. 2.4, A_i are activations that are coming from the learned W_a matrix. So these activations are turned into probabilities to highlight the features more and then by pointwise multiplication of probabilities and the original output of the network, the attended output which is more refined than the original network output is produced. The name additive is due to potential addition in producing A_i, also additive attention is known as Bahdanau attention. In additive attention originally, two factors (sometimes Query and Key, some other tasks it is Encoder and Decoder) are added. For example, In Eq. 2.4, the original formulation of calculating attention

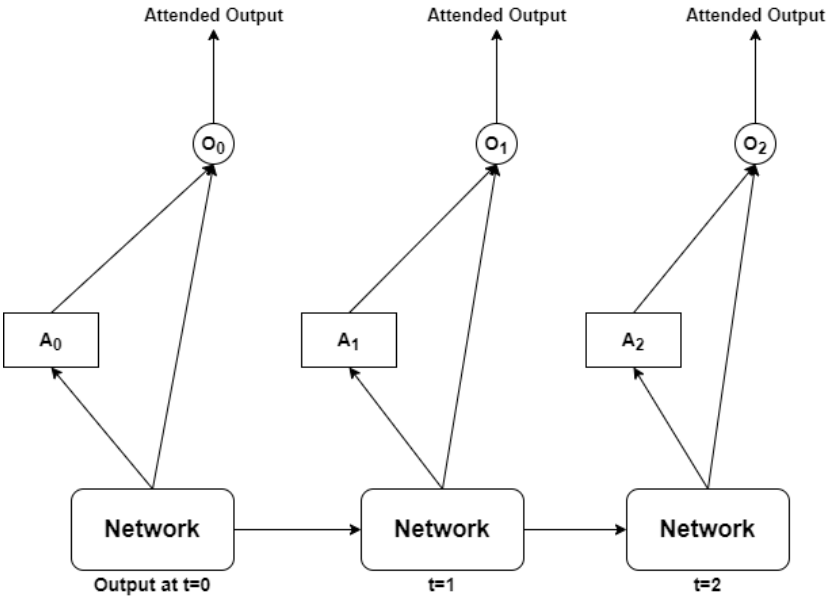

Figure 2.5. Dot-Product Attention Example.

activations should be $A = tanh(W_Q Q + W_K K)$. However, the example shows the one input type so there is no Query and Key, the hidden is directly passed.

$$
\begin{aligned}
A_i &= tanh(\boldsymbol{W_\alpha} t_i + b) \\
v_i &= \frac{exp(A_i w_i)}{\sum_j exp(A_j w_j)} \\
o_i &= v_i t_i
\end{aligned}
\tag{2.4}
$$

Another attention type is scaled dot-product attention introduced in (Ashish Vaswani et al., 2017). This mechanism gets Query (Q), Key (K) and Value (V) matrices and does a mapping from Q and K to V. In its usage in the project and the original paper, the incoming vector to attention mechanism is multiplied by W_Q, W_K, W_V matrices to create query, key, and value transformations. After that, the process shown in Figure 2.6 starts and query-key matrices are used to create attention scores and by multiplying it with value matrix the highlighted output is produced. Until here everything is the same as classic dot-product attention. The "scale" part of the scaled dot-product attention comes in Query and Key multiplication. Attention scores are calculated by softmax with multiplciation of Q and K matrices. However, as shown in Eq. 2.5, scaled dot-product attention scales this

attention scores with a division operation. $\sqrt{d_k}$ is the scaling factor and gives the name of scaled to the mechanism. Calculation steps can be seen in Eq. 2.5. The purpose of this scaling is that in dot-product, if we assume that Q and K has 0 mean and variance 1 and independent from each other, then $Q \odot K$ has 0 mean d_k (dimensionality of k). Since this multiplication grows with the dimensionality of K, scaling factor normalizes it because without this, with high dimensions the multiplication could punish softmax function to smaller values.

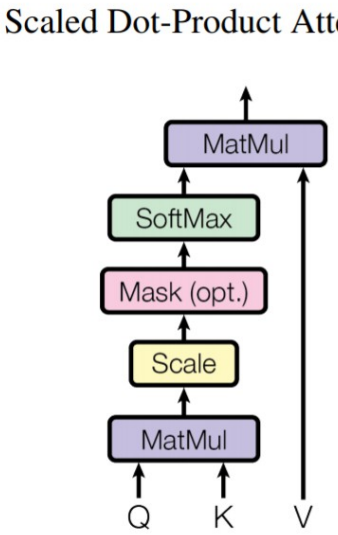

Figure 2.6. Scaled Dot-Product Attention Example.

(Source: (Ashish Vaswani et al., 2017))

$$softmax(X) = \frac{\exp X}{\sum_i \exp X_i}$$
$$ScaledAttention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.5}$$

Where $\sqrt{d_k}$ is the dimension of keys.

Multi-head attention is an attention with the best practice of grouping multiple inputs into one big information vector. There might be multi-channels in neural networks, instead of calculating one by one, multi-head is able to get multi-input and calculate attention then create one final matrix which includes valuable information from the incoming inputs. The process is shown in Figure 2.7. h number of layers are grouped into one big

matrix, and by doing big matrix multiplications on this matrix, one can calculate attention of all h inputs with scaled dot-product attention. Here one additional operation is multiplication with matrices of W_i^Q, W_i^K, W_i^V. With this operation, keys, values, and queries are projected into lower dimension, this is the only purpose. After getting output from the scaled dot-product attention, output of one of h heads (here head represents inputs) is computed. After generating all attentions for all heads, they are concatenated. To project the matrix size into output, these concatenated inputs are multiplied with W^O (just a projection).

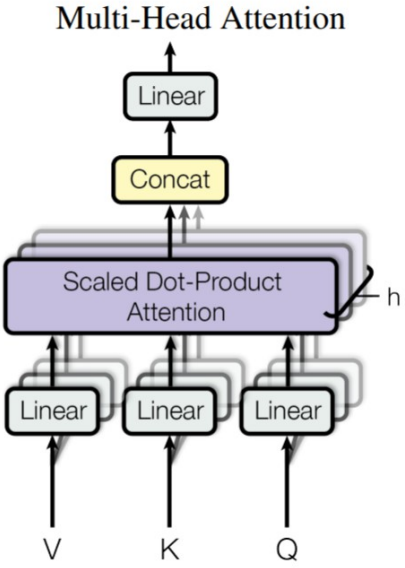

Figure 2.7. Multi-head Attention Example.

(Source: (Ashish Vaswani et al., 2017))

$$
\begin{aligned}
&head_i = ScaledAttention(QW_i^Q, KW_i^K, VW_i^V) \\
&MultiHead(Q, K, V) = Concatenate(head_1, head_2, ..., head_n)W^O
\end{aligned}
\tag{2.6}
$$

2.5. Transformer

Transformer architecture is proposed by (Ashish Vaswani et al., 2017). It is composed of encoder and decoder parts that consist of multi-head attention and linear layers.

The detailed structure is shown in Fig 2.8. The power of transformer comes from its attentive perspective. It neither uses convolution nor recurrence, only attention on a current batch. Although it looks like linear FCNN at first sight, because of high number of parameters it learns much more features and thanks to its self-attention most of the features are valuable ones. So architecture learns many distinctive features.

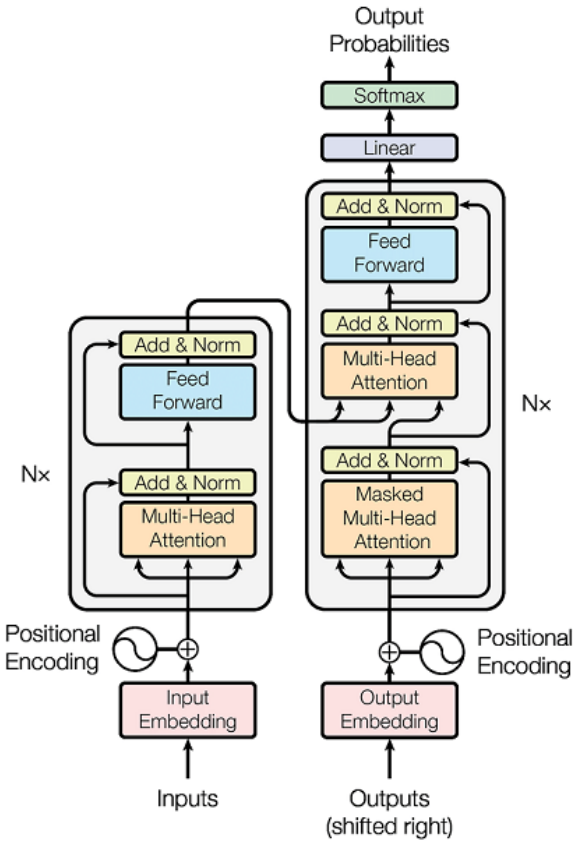

Figure 2.8. A Detailed Architecture of Transformer.

(Source: (Ashish Vaswani et al., 2017))

Models starts with turning words into embedding vectors, by using learnable embedding matrix, then positional encodings are added. This model uses no recurrence so position of the words are needed to be fed to distinguish their order in the sentence, to satisfy this, positional encoding is used. It is just a sinusoidal signal which identifies which comes after which. After that the final embedding is passed into encoder. Operations and flow of tensors are shown in detail in Figure 2.8. N number of encoder and decoder blocks are stacked on one another. Since the original form of this architecture is used for Neural Machine Translation (NMT), in the task of contextual word embedding or cloze

task, decoder part is not used by most of the models. In this project, it is not used either.

CHAPTER 3

RELATED WORK

Research on semantic representations of words is both fundamental and a hot topic in NLP. There are several models approaching in different views to the problem. First of all, there was global vectors where words and represented by only one vector of hand-crafted features. After that, neural networks got popularity and several studies published to make neural networks create more robust word vector, but these were one vector for each word as well. Then contextualization got more popularity and researchers studied on word vectors that changes in context more and combined several different neural networks model with this approach. On the other hand, some researchers focused on not just getting more refined meanings in vectors but also making them able to include syntactic features as well. To get those syntactic features inside the word vector, they had a view of character level conception.

3.1. Character and Word Level Combination Models

There are two different levels of meaning that a word contains. One is word level meaning which represents an object from the real world and the other one is meaning coming from the structure of the language. Suffixes and morphemes are examples of these structural elements. For example, word *worker* has a meaning of "person who performs a job", however this word is constructed from two parts *work* and *-er*. While *work* holds the real world binding of the word, the suffix *-er* indicates that this job is done by someone. While exact translation of *work* can be done between most of language pairs, the meaning coming from *-er* is language specific. Thus, investigating parts-of-words to address the meaning of a word is essential to get the true function of a word. Previous models (Tomas Mikolov et al., 2013; Jeffrey Pennington et al., 2014) consider solely the whole word rather than its structure. However, a model which also learns morphemes

besides a whole word could understand out-of-vocabulary words and would be able to capture both semantic and syntactical features.

Earlier methods focused on getting character level information from a feature set and use it as a base representation to learn the word level meaning. (Andrei Alexandrescu and Katrin Kirchhoff, 2006) processed characters of a word as a set of features by feeding them into multi-layer perceptron and gather a word representation. (Hinrich Schütze, 1993) for the first time used 4-grams as character level features then with summing those 4-grams, he got word level vectors. FastText model (Piotr Bojanowski et al., 2016) is the first study that combined character and word level information by using bag of n-grams with Skip-gram (Tomas Mikolov et al., 2013). Bag of n-grams formed the initial representation of a word and word level representation is learned by Skip-gram. This approach outperformed previous state-of-the-art and is able to process out-of-vocabulary words.

Recent methods use the same neural network to learn both character and word level representations. (Kim Yoon et al., 2016) is the first all-neural method to combine character and word level information into one vector. After that, several studies (Matthew Peters et al., 2018; Rafal Józefowicz et al., 2016; Huadong Chen et al., 2018; Dongyun Liang et al., 2017) adopted the same idea. The general method for these combination models can be described as processing character level with a CNN and passing the representation learned by the CNN to a word level model which is a LSTM. The most successful model for this method is ELMo (Matthew Peters et al., 2018). ELMo uses CNN for character-level since it captures local patterns like syntax. Then it uses highway networks (Rupesh Kumar Srivastava et al., 2015) to reshape and pass the learned syntactic features to a word level model. For word level, Elmo uses LSTM. LSTM and GRU units are very successful in resolving temporality, that's why they are used in word level meaning extraction. In this way ELMo achieves to create a word embedding that implicitly includes both syntactic and semantic information.

3.2. Contextual Word Vector Models

Words might have different meanings in different contexts. As exemplified before, the word "bank" could hold different meanings with respect to its context. The first approach to the problem of identifying words' meanings in their context (senses) was WordNet (George A. Miller, 1995). It is a graph where words can be grouped into conceptual clusters. After WordNet, word sense disambiguation, supervised way of finding different meanings of a word, gained popularity. Several studies (Rada Mihalcea and Ehsanul Faruque, 2004; Kenneth C. Litkowski, 2002; Rada Mihalcea and Andras Csomai, 2005) which use WordNet to disambiguate word senses are published. With the rise of neural models, researchers developed models to disambiguate senses without using any external knowledge base like Wordnet. In other words, they just used unlabeled text corpora to infer senses from the context. Several papers (Dayu Yuan et al., 2016; Minh Le et al., 2018; Mikael Kågebäck and Hans Salomonsson, 2016) are published and outperformed WordNet- based methods.

In 2018, the ELMo model's success introduced the concept of contextual embedding or token embedding. Contextual embedding approaches do not learn multiple different vectors for the same word but they give an embedding to a token (a token is any word in a sentence). Here models process the context of a token and create a representation with respect to that. The solution of "word has different meanings" problem is shifted from word sense disambiguation to the creation of token-based contextual word representations.

3.2.1. Transformer Based Methods

Before discussing transformers, one should mention the attention mechanism which is a building block of a transformer architecture. Attention (Dzmitry Bahdanau et al., 2014) is a mechanism whose objective is to learn the most effective features. Before its usage in transformers, it is used to highlight the most discriminative features in the learned representation layer of a neural network.

The study of (Ashish Vaswani et al., 2017) introduced a new model called transformer. Transformer is composed of encoder and decoder parts, each of which is made up of several stacked attention layers. Transformer uses attention not to highlight features but to match key-value vector pairs given a query. After the success of transformer model in neural machine translation tasks, (Jacob Devlin et al., 2018) introduced BERT that took the transformer approach further to create a comprehensive language model. BERT adopts the transformer architecture with the Cloze Task (Wilson L. Taylor, 1953) as a training objective and use big data from several resources with longer training hours. As a result, BERT model outperformed most of GLUE benchmark (Alex Wang et al., 2018) tests with a high margin. Although it had great result, BERT lacks of morphological information. The only morphological information source is WordPiece (Yonghui Wu et al., 2016) embeddings and byte-pair encoding (Rico Sennrich et al., 2016). However all of these require intensive analysis on the target language. Although this analysis is performed in English in (Yonghui Wu et al., 2016) paper; underrepresented languages, like Turkish, lacks of these analysis.

After BERT model's huge success, several extensions are introduced with a similar approach. RoBERTa (Yinhan Liu et al., 2019) model increased data and training hours and made a small change to training objective by removing next sentence prediction and achieved better results than BERT. GPT (Alec Radford, 2018) (Although GPT is released months before BERT, it was only a preprint) and its successor GPT-2 (Alec Radford et al., 2019) use transformer with more parameters and data than BERT and achieve a broad range language model. By broad range language model, it is meant to be able to understand formal, informal, definitive, artistic, smiley-intensive texts better. ALBERT (Zhenzhong Lan et al., 2019) model focused on optimizing BERT to require fewer resource and fewer training time to achieve similar or better results.

Some researchers are sceptic about the success of transformers. The study of (Sarthak Jain and Byron C. Wallace, 2019) investigated attention mechanisms to examine whether the success can be attributed to them. They claim that attention mechanism is not robust enough to create an explanation because different inputs can yield same attention (means attention scores might be produced arbitrarily) and feature importance metrics does not meet the output of attention highlights (feature importance metrics does not

show that highlighted features by attention is not distinctive than the others so much).

3.2.2. LSTM Based Methods

LSTM (Sepp Hochreiter and Jürgen Schmidhuber, 1997) is a recurrent neural network architecture which solves the vanishing gradient problem of vanilla RNNs (Jeffrey L. Elman, 1990). Study of (Yoshua Bengio et al., 2003) is the first work that proposed RNNs for building a language model. Several studies (Bryan McCann et al., 2017; Stephen Merity et al., 2018; Matthew Peters et al., 2017) used LSTMs in language modeling, however the ELMo (Matthew Peters et al., 2018) model created a huge impact and gave successful results.

ELMo model combines the approach of character-level language modeling with a word-level language model. In a character level LM, the objective is to predict the next character in the sequence. This objective helps to capture morphological features (suffixes, compound word etc.), but it has a poor performance in addressing semantics. The reason behind can be explained two-fold: Character based sequences are longer than word level ones and longer the sequence is a higher probability of forgetting. The second reason is that in word level every token holds a meaning on its own but in character level only the combination of characters can create a meaning. In word-level LM, the target is to predict the next word in the sequence. Since the sequence length is shorter and each element holds a meaning on its own, this approach is better to learn meanings. However, the disadvantage of word level models is that they do not capture the morphological structure. So words *paper* and *papers* are two different independent entities for word level models, although they are related to each other in natural language. ELMo combines these two approaches and creates one contextual word vector which holds both semantics and morphological information.

CHAPTER 4

METHODOLOGY

The BERT model and the proposed model have lots of additional features that other language models do not have. This chapter explains our intuition behind the proposed model, how we built it, and the evaluation framework.

4.1. Proposed Model

The proposed model is a combination of BERT (Jacob Devlin et al., 2018) and ELMo (Matthew Peters et al., 2018) models. Model details will be provided along with BERT and ELMo models in order to describe the architecture with its justification.

4.1.1. Intuition of the Model

As mentioned earlier in Section 1 and Section 3, there are word level and character level language modeling. These two perspectives have their advantages. Character based models focus more on morphology and syntax, but relatively poor on capturing meaning of the word. On the other hand, word based models are better in capturing semantics but agnostic in syntactic level. Additionally, there are also hybrid models that combines word and character level information into one word vector. Challenge of combining is a fundamental problem of information transfer actually.

In the proposed model, character level and word level perspectives are combined. It can perceive at the character level and produce output at the word level. We inspired the idea from the ELMo model where the authors used CNN to process characters then by building word vectors from the CNN, they processed those words to create contextual word embeddings.

After the huge success of ELMo, a Transformer based language model, BERT

is proposed. This model outperformed ELMo in many tasks of the GLUE Benchmark. However, BERT is a word level model and the only syntactic information comes to it from the power of WordPiece embeddings. This WordPiece style morphological analysis is not complete in many languages. Thus, to adapt BERT to another language requires intensive analysis on the morphological structure of that language. To get rid of this requirement and enrich the contextual word representation of BERT, this thesis proposes a new model, named BERT-ELMo, which uses a CNN based character level structure under the BERT architecture.

4.1.2. Architecture of the Proposed Model

ELMo model uses CNN to capture characters and uses two highway networks and dimension projection after that to capture character information and compress it into a vector to pass upper word level models. The architecture can be seen in Figure 4.1. ELMo uses LSTM to create contextual word vectors, but in the proposed model, BERT will be used replacing LSTMs.

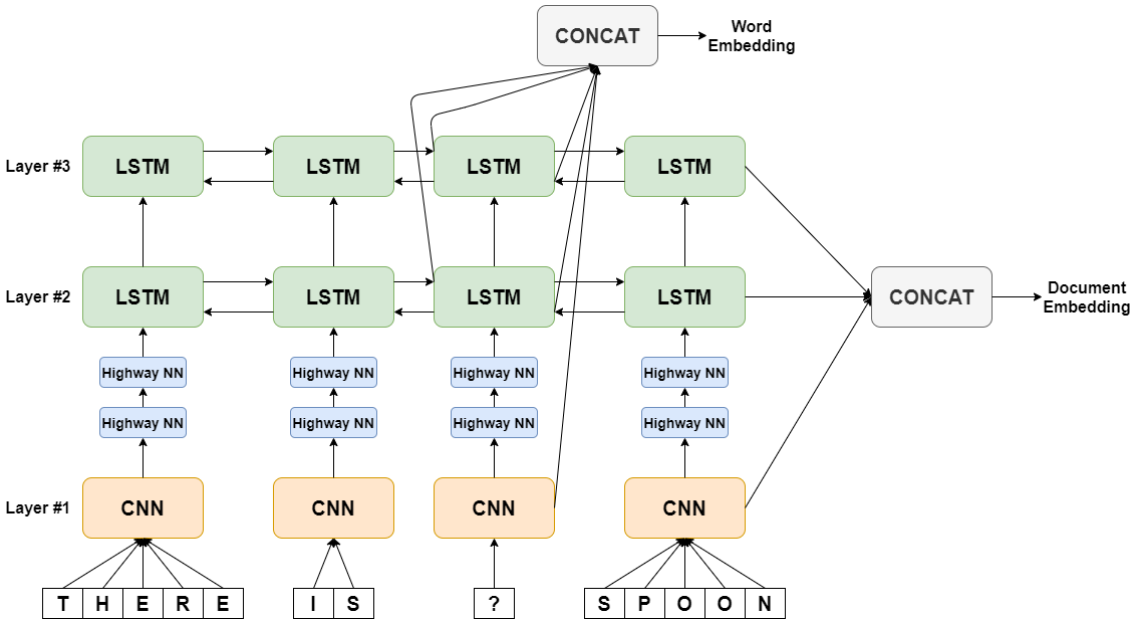

Figure 4.1. ELMo (Matthew Peters et al., 2018) Architecture.

Proposed model takes operations below LSTM layers from ELMo model. It cre-

ates input word vectors for BERT to process them and create contextualized word embeddings and document/sentence embeddings. The embedding layer of proposed model is shown in Figure 4.2. Each word vector will be fed into BERT and they are not temporal but simultaneously in the batch.

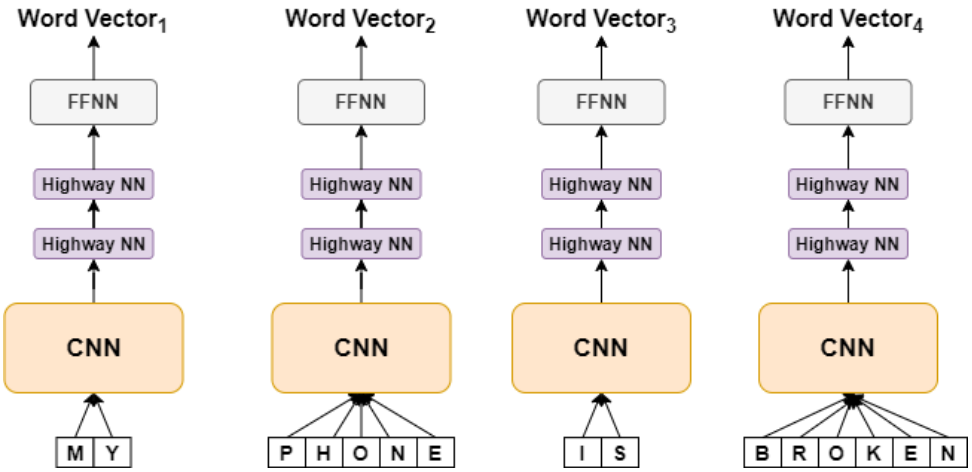

Figure 4.2. Embedding Layer of the Proposed Model.

BERT model uses Encoder part of the Transformer Architecture (Ashish Vaswani et al., 2017). A generalized look to BERT and its encoder modules are shown in Figure 4.3. This stacked encoders capture more contextualized information as we go higher in the stack.

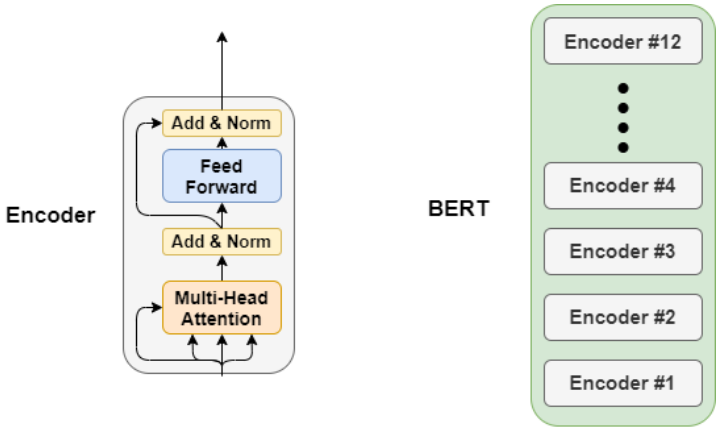

Figure 4.3. BERT (Jacob Devlin et al., 2018) Architecture.

Proposed model combines the embedding layer and BERT which are explained

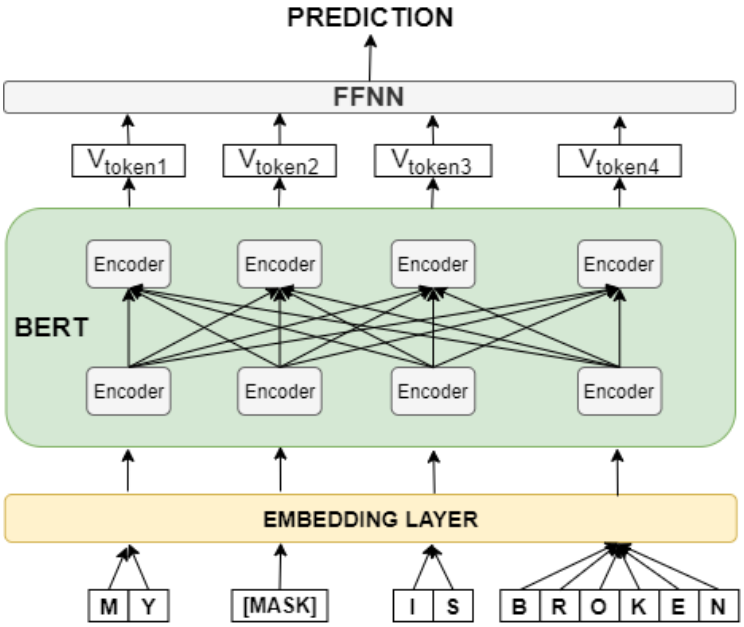

Figure 4.4. Proposed Model.

above. For visual purposes, only two encoder layers are shown in Figure 4.4. However, model has 12 encoder layers. The power of BERT comes from its attention-only structure and non-temporal process (does not need to wait word from 1 to N to finish in order to process $N + 1$). But it captures temporal information from positional embeddings, so it gets the necessary information without processing one by one . It does not concern with forgetting or misjudging previous data, model just takes all sequence at the same time and each encoder is connected to every other in upper layer. So, for each token (or word) encoder do processing according to its own token and any other token. This feature is the root of contextualized embeddings in BERT and proposed model. With the proposed model, BERT gains power of learning syntax in all languages, and able to produce results for out-of-vocabulary words as well.

4.2. Training Procedure

In neural networks, training procedure is very important. Because model performance, especially that of BERT, is sensitive to initial weights, objective function, and optimizer. Training specifications and training time are crucial to replicate the model.

4.2.1. Cloze Task

In classical language modeling, as described well in (Yoshua Bengio et al., 2003), the task is to predict the next word given the set of previous words. It is straightforward to think like that because when we speak we go incrementally and produce next word at each time step. Lots of language models are trained in this fashion and the most successful among them is the ELMo (Matthew Peters et al., 2018) method. It is a bidirectional language model, so starts to process from start to end and end to start. So seems like two LSTMs in one layer, one is going to W_1 to W_t forward and the other one goes from W_N to W_t backward, if we assume that there are N words. Generally, after processing to W_t from both sides, the hidden outputs of backward and forward are concatenated to create the final representation for the word W_t.

Our proposed model BERT-ELMo is trained with the cloze task like BERT. Cloze task is a problem to read whole text and guess the missing words in it without any direction. First the whole text is fed into the system, then some words are taken out and those positions are marked to indicate that there is a missing word (it is "..." in English exams and [MASK] in this thesis). BERT and our method use this training objective. Since whole context is visible and processed before the prediction of masked word, result is more contextual.

Two different sentences are selected and model masks some words (with a certain probability of masking). After processing the input, models produce a probability distribution of possible words for the masked place, and with the one-hot notation (length is the same as the length of vocabulary), cross-entropy loss is calculated to train the model. A higher rate of masking would result in information loss (if all words are masked, then no prior information is fed to the system), but a lower rate causes a small number of training samples, because networks learn from the masked words.

BERT introduces its own input data format to separate two sentences and some tokens. The sample input with masking can be seen in Figure 4.5. [CLS] tag indicates the start of a sequence, also its activations in the final hidden layer is accepted as a sentence or document embedding. [MASK] tokens indicate the location of masked words and the predictions on these locations create learning signals. [SEP] token stands for the end of

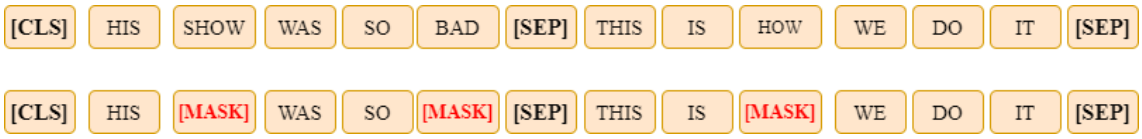

Figure 4.5. An Example of Cloze Task in BERT Notation.

a sequence. Here it is called as sequence not a sentence because the time complexity of BERT is quadratic to the sequence length. So, to get rid of huge sentences to create very long waiting times, the sequence length is fixed, and to use this sequence length effectively, if a sentence finishes before the length, rest is filled with some other sequences with random sampling (not just arbitrary words but a part of another sequence to fill the empty area).

Additionally, BERT uses Next Sentence Prediction (NSP) as another task. It tags words in the first sequence with A and the ones in the second sequence with B and network is forced to learn where the first sequence ends and where the next one starts. Network creates embeddings at each word for this task and this creates a loss as well from the ground-truth sentence tags. In the original paper (Jacob Devlin et al., 2018), authors claim that NSP is crucial to train network better and support the claim with ablation tests. Total loss is calculated by the sum of masked language model loss and NSP loss.

4.2.2. Training Settings

Training specifications are essential to replicate a result. To be more transparent about the training and results of models, specifications are listed. Source codes[1] of all models are publicly available. The common parameters for all models are listed below:

- Masked Language Model Probability (0.15): It is a probability of one word to be masked. So 15 out of 100 words will be masked.
- Max Sequence Length (96): Since the sequence length must be fixed to run models efficiently, 96 is chosen as length. The reason behind is that 8 batch size and 96 sequence length are the optimum values when the memory of a GPU is considered.

[1] https://github.com/polatbilek/bert , https://github.com/polatbilek/bert-elmo

- Batch Size (8): As mentioned earlier, 8 batch and 96 sequence length are the optimum.

- Encoder Specifications: The number of encoder stacks is 12. Attention dropout probability is 0.1. Hidden activations inside encoders are GELU (Dan Hendrycks and Kevin Gimpel, 2016) and hidden dropout probability is 0.1 and hidden vector size is 1024. Number of attention heads is 8.

- CNN Kernels: For character level perception of BERT-ELMo, several different sized CNN kernels are created. The kernel size and numbers are: [(1, 32), (2, 32)], (3, 64), (4, 128), (5, 256), (6, 512), (7, 32)]. The kernels are executed on full word so one's output is not other one's input. And the first number of the tuples are the kernel size and the second number is the number of kernels for that layer.

- Learning Rate ($2e^{-5}$): Learning rate is one of the most important hyperparameter of neural networks especially BERT-like networks. The original paper's suggestion is used as the learning rate, because running time of models were approximately one month, so no hyperparameter optimization is done.

- Training Steps (6000000 and 500000): BERT-ELMo model is trained in 500000 steps because of time limitations. So to make the model comparable, the BERT model (this model will be named as BERT-500K) is trained in the same number of steps as well. In addition, BERT model is also trained in 6000000 steps (this model will be named as BERT-6M) to see the capability of the model when it is fully trained. Number of training steps in an epoch is calculated as number of data per batch size.

- Warm-up Steps (0.1 ratio): Warm-up steps are training with increasing learning rate to increase the model's search space. 0.1 rate is used to calculate the number of warm-up steps. For 500000 training steps, there are 50000 warm-up steps so it makes 550000 steps to make the model finish the training.

- Training Environment: BERT trainings are done on a local server with 32 GB RAM, Nvidia TITAN V GPU (12 GB GPU RAM) in one month for 6000000 steps

training and 2 days for 500000 steps training. For BERT-ELMo model, because of technical problems runs could not taken in the same local server. Instead, they are taken on Amazon Web Services with g3.4xlarge instance which has 128 GB RAM and Nvidia Tesla M60 GPU (8 GB GPU RAM) in 20 days for 500000 steps training which is the only training. All works are coded in python programming language with Tensorflow framework in linux operating system.

4.3. Language Model Training Dataset

To show the model's ability of capturing morphological information, Turkish language has chosen as the target language. Model is trained on the Wikipedia dump [2] [3] of Turkish articles. These articles are preprocessed to BERT's original format and become lowercased. Other than these, no manipulation is performed on the text. The text contains some words from other languages as Wikipedia articles mention something from other languages.

In BERT-ELMo model, the dataset is used as it is and vocabulary is a mix of some frequently used Turkish words and the words that have more than 10 times occurrence in the dataset. So words are broken down into characters, Character-CNN part processes those characters, and create word embeddings for each known word. For unknown words, [UNK] label is used. Sentences are tokenized by just splitting from space.

In the BERT model, the dataset is not modified but the vocabulary is. Because to use BERT as original, one needs WordPiece word tokenization. WordPiece splits a word into its stem and suffixes. For example; word *holding* is divided into *hold* and *ing*. Here tag stands for where the suffix sticks to the stem in original text. With this feature BERT knows that *holding* has two parts and creating two different embeddings for those. By averaging all parts of a word, one can find the true embedding of a word. To achieve this on Turkish, firstly, an intensive search has been done on studies about morphological analysis on Turkish Language. A graduate thesis (Orhan Bilgin, 2016) and a Github project (Ahmet Aksoy, 2017) have been found that have already done Turkish stem and

[2] Collected in January snapshot of Turkish Wikipedia

[3] Dataset is downloaded from https://tr.wikipedia.org/wiki/Vikipedi:Veritaban%C4%B1_indirme

suffix words analysis. Stem and suffix dictionaries are gathered and unioned into one big vocabulary of Turkish stem and suffixes.

4.4. Evaluation

Testing the hypothesis by an experimental evaluation and analyzing the results are the essential part of the scientific process. To evaluate a hypothesis; qualitative or quantitative methods could be used. Qualitative methods are the techniques that control whether the hypothesis matches with the intuitional expectations through examples. Rather than numbers and ratios, qualitative evaluations are based on explanations and understandings. Quantitative methods, on the other hand, test the effectiveness of models through objective measures. These methods introduce metrics such as accuracy, F-1, BLUE score etc. As a rough analogy from software engineering, qualitative methods could be called as validation and quantitative methods could be called as verification for the problem domain.

4.4.1. Qualitative Evaluation

In this section, to test the quality of the word embeddings produced by the model, word similarities and/or analogies will be evaluated and some conclusions will be made.

4.4.1.1. Word Analogy

Word analogy task is a problem of capturing the analogy between a pair of words. A pair of words and a query word are given and the task is to predict the pair of the query word given the analogy. To clarify, if you think a pair like (Turkey-Ankara) then one can see that there is a country-capital relationship, then feed (France-?) to the system and let it predict the corresponding word (which should be Paris of course). Although the task is introduced as a task of language understanding of models in (Tomas Mikolov et al., 2013), the first usage was in (Peter D. Turney et al., 2003) study. Authors used multiple-choice

analogy questions from SAT exam to test their models. (Tomas Mikolov et al., 2013) study made it more popular thanks to a refined, broad dataset that they introduced. After that paper, this task became a general task in the evaluation of word embedding models. However, in the evaluation of proposed and compared models, available datasets are not suitable for a quantitative analysis. Because these datasets provide only two tuples (where each tuple contains a relationship). As all the models in this thesis are contextual, a context paragraph is needed as well to create word embeddings. Thus, this analogy task will be used qualitatively and we will show how well the model performs on these analogies. To make analysis more comparable, all models will be fed by the same paragraphs and same analogies will be queried.

To make this analogy work, some vector operations are performed. For example, in capital-country analogy:

$V(target) = V(Ankara) - V(Turkey) + V(France)$

$TokenList[argmax(cosine\_similarity(V(Word_i), V(target)))] = Paris$

where index i scans all the words in the context.

$V()$ represents vector of the target word. Here the analogy is if what Turkey represents is taken out from Ankara, what left is bare capital features. When features of France are added to this vector, it will be $Capital + France$. Here $Paris$ result could be expected. Since the proposed model does not produce words that are not in the sentence, not the whole vocabulary but the words inside the sentence will be compared with the target vector. In this thesis, capital-country, plural-singular, king-queen analogies will be tested. Also $p-value$ will be provided to show the statistical significance of the similarity among others.

4.4.1.2. Contextuality of Words

BERT model (and proposed model) is good at creating contextual embeddings. In this task, the quality of contextual embeddings will be tested in a polysemous setup where a word has different meanings in different contexts. The word "yüz" could be given as an example. It may hold for a meaning of number or a meaning for front of head

depending on the context. The contextuality of the model will be tested using the words "yüz", "dil", "sol" to show whether it assigns different vectors for different contexts. In this task, cosine similarity will be measured between target word ("dil", "sol" or "yüz") from target context and compared contexts. There is one target context for each word and the compared contexts are paragraphs with the same word but different meanings. So the word similarity between target word from one meaning with two others from different contexts are measured. In this task, lower the similarity score means more contextual the model is since comparisons are on different meanings of the same word.

4.4.1.3. Morphological Feature Learning

In Turkish, suffixes might add several different types of meaning to a word. Some suffixes make the word negative, others turn verbs into nouns and these suffixes may be chained as well. In this test, models will be tested on their ability of capturing morphological features. One important thing to mention is that before training BERT, some morphological analysis on Turkish is gathered and put in the vocabulary of BERT. So BERT is aware of suffixes to some extent. This means that BERT will capture some morphological structures, which are predefined in its vocabulary. This study's contribution on the morphological feature capturing is that proposed model is able to capture any morphological structure without predefining it (due to character-aware ELMo layer) and it is believed that it will be more successful than BERT on capturing morphological features.

To test this suffix effect, we use three types of suffixes in Turkish. One is first person and second person ("ben" and "sen") possessive suffixes like "kalemim - kalemin". Without using word "ben" or "sen" in the sentence and by only changing last character, we expect the model to learn the meaning shift from "ben" to "sen". This shift will be tested by the cosine similarity between "kalemim - kalemin" and "ben - sen" word vector pairs. Other than this possessive suffix, no other change will be done on the sentence. Another one is negativity suffix in Turkish. For example, "çizmedim - çizdim". Here the suffix "-me" adds negativity and this negativity will be tested in the sentence with the same method used in contextuality test. The last effect is tense suffixes, for example "geldi -

gelecek". Here different from the English language, word gets several different suffixes. It is not only "-ing" and "-ed" but could get different time suffixes with some integration suffix as well. To test this, two sentences with the same words but their verbs in different tenses are created. Additionally, two more sentences, where one of them contains the word "yarın" the other one contains "dün", are fed into the model and embeddings are gathered. Here we expect the vector of $geldi - dun + yarin$ to be similar to $gelecek$.

4.4.1.4. Out of Vocabulary Word Evaluation

In this part, BERT-ELMo model will be tested on word vectors for the words that were not in the vocabulary of BERT. Because of embedding layer of BERT-ELMo, proposed model is able to perceive all words and generate embeddings for them. These embeddings might not be perfect but being able to process all the words make BERT-ELMo both language agnostic and better self-learner. Unlike BERT, BERT-ELMo is able to get trained in any language if characters of that language is shown in the vocabulary (generally it is easy to generate all characters). For training BERT, a morphological pre-analysis is required for better training and it is done in compared models in this study. But for BERT-ELMo, no pre-analysis is needed, it is trained as it is and able to process all words without any unknown. Also this supports the self-supervised method of the masked language model training because it is all self-supervised thanks to embedding layer part.

To test this ability of BERT-ELMo, three target words are chosen and two contexts are gathered for each. And there are two versions of these two contexts. One version is the original paragraph, the other one is exactly the same paragraph but target words are replaced with the word "Wug". This is called as "Wug" test which was firstly conducted in (Jean Berko, 1958) to learn the language acquisition of children. In this context, the "wug" test is conducted to check whether the model assigns a similar vector even though word is replaced with "wug". For example, if our target word is "picture" and one of the context is "I love taking pictures. This is my favorite picture." then the wug-transformed context is "I love taking wugs. This is my favorite wug.". Each target word is an unknown word for both BERT and BERT-ELMo vocabulary. So even the original words are out of

vocabulary. Because of BERT-ELMo's character level perception, it is able to create embeddings for those words. BERT-ELMo has a word vocabulary but it is only used to predict the masked word, all unmasked words are processed even though they are not shown in vocabulary. So word vocabulary is present only to be able to make a prediction from vocabulary id of the word. So, to test this feature, words that are not in vocabulary are selected with two contexts and two more contexts are created out of these by changing target words with "wug". The context paragraphs could be found in Appendix D.

4.4.2. Quantitative Evaluation

To test the robustness of language models, generally performance in subtasks is calculated. In this thesis, question answering subtask is selected to test the robustness of models. All the examples and contexts are gathered before testing the models. No adjustments are made to contexts, only two contexts are replaced, which contained highly [UNK] labels since models were not able to throw any signal for them.

4.4.2.1. Question Answering Dataset

The Turkish question answering dataset is created by earlier projects. It has exactly the same format with the SQuAD dataset (Pranav Rajpurkar et al., 2016), it contains context text, and questions derived from it. The task is to find answer span in the context text. The questions and context are gathered from Wikipedia articles in this project. So the language use of the subtask (Question Answering) is not tested here since both training of language model and question answering are done on the same source.

The Turkish question answering dataset has 9200 questions in its training set and 892 questions in test set. The longest answer's length is 40 words and all contexts have varying number of questions and text length. No preprocessing is done on these texts other than lowercasing. Some examples from the dataset can be seen in Table 4.1.

4.4.2.2. Evaluation Method

Here the training objective is identifying the answer span correctly. As the location of all tokens are already known, basically network tries to produce two numbers, starting and ending location, and compare it with the ground-truth. Cross-Entropy Loss is used to train the network. The loss is computed between one-hot encoding of ground-truth start and ending position and probability distribution of start and ending position from predictions. Total loss is computed by averaging start and ending position losses.

Also to be able to process this dataset, input notations are modified. In question answering input style, the first sentence is question and the second is the context paragraph. An example input would be like:

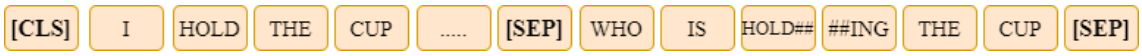

Figure 4.6. An Example Question Answering Input Sequence.

The output of the final layer of the models are linearly transformed to be comparable with one-hot ground-truth position vectors. So, with given input type as above, then doing linear transformation to the final hidden layer output and finally by calculating cross-entropy loss, the network is trained for the task of question answering. Prediction process for Question Answering Task is shown in Figure 4.7.

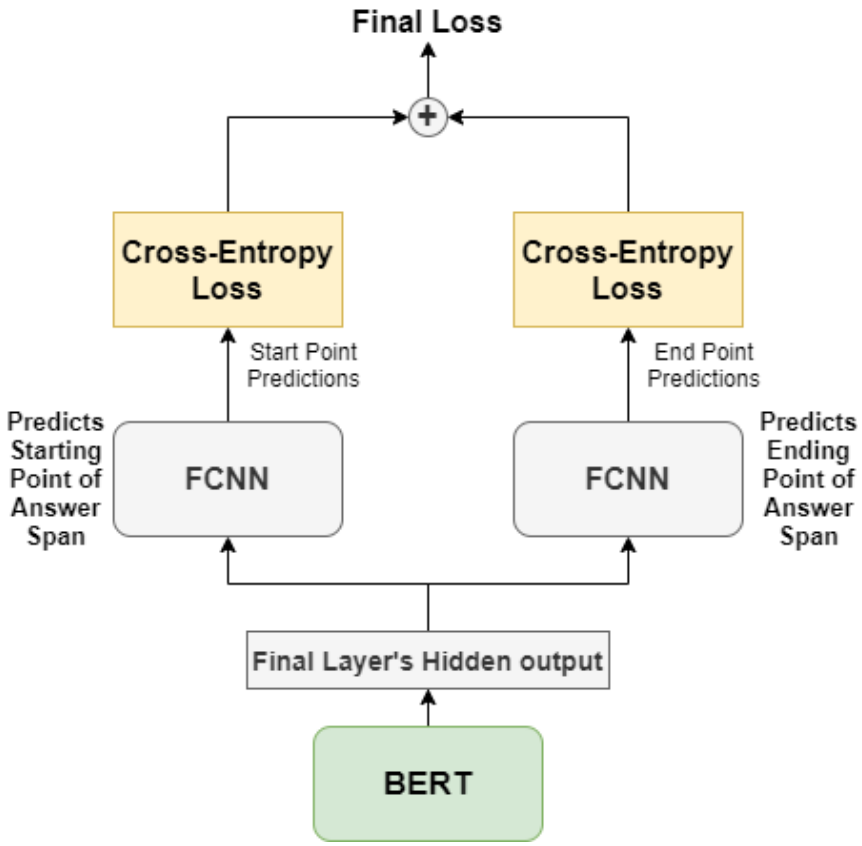

Figure 4.7. Question Answering Prediction Network.

Table 4.1. Sample Paragraphs, Questions and Answers from Turkish QA Dataset.

Paragraphs	Questions	Answers
Adana Bilim ve Teknoloji Üniversitesi, şu anda 3 küçük yerleşkede eğitim vermektedir. Mühendislik ve Doğa Bilimleri Fakültesi Yeşiloba Yerleşkesinde, YADYO Kurttepe Yerleşkesinde ve geri kalan fakülteler Ziyapaşa Yerleşkesinde eğitim vermektedir. 2017 Eğitim yılı başlangıcında Sarıçam Kampüsünde eğitime devam edecektir.	Adana Bilim ve Teknoloji Üniversitesi 2017 Eğitim yılında nerede eğitime devam edecektir ?	Sarıçam Kampüsünde
	Adana Bilim ve Teknoloji Üniversitesi'nin Mühendislik ve Doğa Bilimleri Fakültesi neredeydi ?	Yeşiloba Yerleşkesi
	Adana Bilim ve Teknoloji Üniversitesi kaç tane yerleşkede eğitim vermektedir ?	3 küçük yerleşkede eğitim vermektedir.
Prof. Ümran İnan Dört ders kitabının yazarıdır. Bunlardan ikisini kardeşi Aziz İnan ile birlikte yazmıştır. Ayrıca Prof. Ümran İnanın hakemli dergilerde 330'a yakın makalesi yayınlanmış ve bu makalelere 7400'den fazla atıf yapılmıştır. Engineering Electromagnetics (Prentice Hall 1998) Electromagnetic Waves (Prentice Hall 1999) Principles of Plasma Physics for Scientists & Engineers Numerical Electromagnetics	Umran İnan yazdığı 4 ders kitabının 2'sini kiminle yazmıştır?	Aziz İnan
	Umran İnan'ın makalelerine yaklaşık kaç atıf yapılmıştır?	7400
	Umran İnan'ın kaça yakın makalesi dergilerde yayınlamıştır?	330
TAI Sivrisinek; TUSAŞ şirketi tarafından Türk Silahlı Kuvvetleri'nin ihtiyacı doğrultusunda üretilen "Rotorlu İnsansız Hava Aracı" (R-İHA). Yapılan ilk denemede 150 metre uçurulan hava aracı, 150km lik bir menzile sahip olacak. Ayrıca Roketsan'nın tasarladığı 8km menzilli, T 129 helikopterinde de kullanılacak olan Türkiye'nin ilk lazer güdümlü füzesi olan "Cirit" de bu hava aracında kullanılmaktadır.	TAI Sivrisinek ilk denemesinde kaç metre uçurulmuştur?	150 metre
	TAI Sivrisinek'i hangi şirket üretmiştir?	TUSAŞ
	TAI Sivrisinek ne tür bir araçtır?	Rotorlu İnsansız Hava Aracı

CHAPTER 5

RESULTS AND DISCUSSION

In this section, results of proposed and compared models are shown and will be discussed. Since BERT-ELMo model's training will last long (~20 days for 500k iteration), a BERT model, named as BERT-500K, which is trained in the same number of steps with BERT-ELMo is created. The fully trained BERT model is named as BERT-6M which is trained in 6 million training steps. This model is created to show the full capability of BERT and scaling factor for BERT-ELMo, BERT-500K model is the actual comparison base for BERT-ELMo.

5.1. Results

From Table 5.1, it can be seen that all models achieved acceptably high accuracy results in next sentence prediction. Next sentence prediction is the training objective of the models where the word that introduces a context shift into another sentence is predicted. Losses do not indicate much on their own but by combining losses with the training loss graphs in Appendix B shows that BERT-ELMo and BERT-500K could still manage to learn more, they are much like underfit due to time limitation in BERT-ELMo's training. Although these models are underfit, they got acceptable results in masked language model objective. Out of the models concerned, as expected BERT-6M outperformed the others. One additional note here is the accuracy difference between BERT-ELMo and BERT-500K. There is a noticeable difference between masked language model accuracy of BERT-500K and BERT-ELMo, even though they are trained in the same number of steps. The reason can be attributed to the difference in the number of parameters. BERT-ELMo both includes all parameters from BERT and parameters from word embedding model (CNN) part. This uneven number of parameters requires different training procedures but the hyperparameters are kept the same for all models to make them comparable,

because there was no chance for hyperparameter optimization.

Table 5.1. Test Scores on Language Modeling Training.

Metrics	BERT-6M	BERT-500K	BERT-ELMo
MLM Loss	0.928	1.678	3.278
MLM Accuracy	0.797	0.669	0.553
NSP Loss	0.029	0.178	0.141
NSP Accuracy	0.992	0.928	0.943

Even though, the training accuracy of BERT-ELMo was lower than compared models, it achieves similar or better results with BERT-500K model in several subtasks. This shows that the proposed improvement of adding CNN based embedding layer preserved the ability of capturing word level meaning as BERT.

5.1.1. Qualitative Analyses

There are four subtasks in qualitative analysis: Analogy, Contextual, Morphological and out of vocabulary analysis. To feed models with input contexts to get embeddings, two contexts needed. Since two contexts would affect each other, we fed first the true context then a meaningless word as second context. Since this meaningless word (this word is "wwgg") will create [UNK] tag for all inputs, it will be just constant noise for all samples. "|||" shows the context seperation. It is like the [SEP] tag which in explained previously. One context is given at each evaluation, so to fill the second context part a meaningless "wwgg" word is used for all samples. Also there are some results with P-values to show the significance of the result. P-values are calculated between cosine similarity score of all words in the sentence. All P-values are under 0.05 which shows that all values are significant. The reason of this the results scores are high while most of the words raise similar cosine scores. The context paragraph for the target words are shown in Appendix D.

5.1.1.1. Analogy Task Results

For the Analogy Task, two different sentences are fed. A meaning analogy is created and operated with word vectors and the resulting vector is compared with the all word vectors of second context. The most similar word is taken shown in Table 5.2 with its similarity score and P-value with respect to the similarity distribution with other words in that context. One additional thing to mention is that in these kind of similarity comparisons, the word that are used to create $query$ vector are discarded for evaluation.

For Gender Analogy:

$V(query) = V(kral) - V(kralice) + V(prens)$

It is expected that $V(query)$ vector is the most similar with $V(prenses)$.

For Plural Analogy:

$V(query) = V(halkalar) - V(halka) + V(kalem)$

It is expected that $V(query)$ vector is the most similar with $V(kalemler)$.

For Capital Analogy:

$V(query) = V(ankara) - V(turkiye) + V(fransa)$

It is expected that $V(query)$ vector is the most similar with $V(paris)$.

Table 5.2. Analogy Test Results.

	Gender		**Capital**		**Plural**	
	Sim.	P-score	Sim.	P-score	Sim.	P-score
BERT-6M	0.397	0.0	0.390	$6\text{x}e^{-69}$	0.625	$3\text{x}e^{-47}$
BERT-500K	0.702	0.0	0.385	$9\text{x}e^{-42}$	0.576	$4\text{x}e^{-61}$
BERT-ELMo	0.606	$1.5\text{x}e^{-216}$	0.371	$9\text{x}e^{-50}$	0.559	$3\text{x}e^{-57}$

For all the results, the most similar word was the one it is expected and the all P-values show that this similarity can be easily distinguished. Even though in gender analogy BERT-ELMo model achieves higher result than others, it gives relatively close results with BERT-500K. This example shows that all the models are able to include fea-

tures that is related with the word. For example, in $V(ankara) - V(turkiye)$ operation, features of $turkiye$ is subtracted from $ankara$ so what is left is features of a capital without any country specific feature. But $ankara$ specific features are still conserved. By adding $V(fransa)$ with this vector, the new direction shift towards more French features. After the first operation, $ankara$ specific features are still conserved so no one can expect the $query$ vector (the result vector of $ankara - turkiye + fransa$) to be exactly similar to $paris$. Here combination of french features with features for a capital results in $Paris$. All models achieved the similar results, and as expected BERT-6M is better than other models since it is fully trained. This results show that BERT-ELMo has as same capability of capturing analogies as BERT.

5.1.1.2. Contextuality Task Results

In "non-contextual" word embeddings, the embeddings of same word would be similar with score of 1 as they would be identically same. In contextual models, a vector of same word may be similar or dissimilar to each other with respect to its context. So vectors of polysemic words differ from context to context as they are in natural language. To evaluate contextuality, same words from three contexts are selected. One context uses the word in a different meaning of it other than rest of the contexts. So there is two different meaning of the word, one context uses one, the other two contexts use the other meaning. For example, "being a body part" meaning of the word "dil" is selected as target and selected one context for it. "Natural language" meaning of the word "dil" is selected for comparison, so that they must be dissimilar to each other. For this meaning two contexts are selected to show that this dissimilarity is not about a coincidence but consistent. In this task, dissimilarity shows that models are contextual enough to assign different vectors to same word. Any result other than 1 is actually shows that models assign different vectors. In order to control that what is the similarity when meanings are same is measured and shown as "Control" column in Tables 5.3, 5.4, 5.5. In "Control" columns, the similarity of the word from target context is measured with same word from another context paragraph where the meaning of the word is same, is measured. Because

these contexts use the same meaning of the word. So target word is tested on context 1 and context 2 for different meanings and control group is the same meaning so that the base should be known. Also the context for target meaning and compared contexts are shown in Appendix D.

Table 5.3. Similarity Results of Word "Dil" for Contextuality Task.

	Dil		
	Context 1	Context 2	Control
BERT-6M	0.683	0.577	0.775
BERT-500K	0.815	0.551	0.836
BERT-ELMo	0.853	0.727	0.882

Table 5.4. Similarity Results of Word "Sol" for Contextuality Task.

	Sol		
	Context 1	Context 2	Control
BERT-6M	0.507	0.565	0.621
BERT-500K	0.704	0.690	0.668
BERT-ELMo	0.610	0.547	0.679

Table 5.5. Similarity Results of Word "Yüz" for Contextuality Task.

	Yüz		
	Context 1	Context 2	Control
BERT-6M	0.815	0.744	0.919
BERT-500K	0.918	0.748	0.948
BERT-ELMo	0.959	0.867	0.977

From the Table 5.3, it can be seen that BERT-6M clearly established contextuality in word Dil's meanings. The similarity from same context (shown as "Control") is clearly higher than other contexts. Even though, they hold for different meaning, it can not be expected from the model to assign nearly zero similarity to the word. That would be inaccurate similarity as well, since it is same word anyway, so they have common features. What an be expected only is the similarity of vectors from different contexts to be smaller than the ones from same context. BERT-6M achieves that for word "Dil".

On the other hand, BERT-500K assigns highly similar vector for context 1. It has still lower similarity than control but they are close enough to say that BERT-500K could not manage to distinguish words from target well. This trend is same in BERT-ELMo. But it assigned even more similar vector for context 2. This results show that BERT, when it is fully trained as in BERT-6M, is able to contextualize words and BERT-ELMo has same capability of BERT because it kept nearly same trend with BERT-500K.

For the word "Sol", BERT-6M and BERT-ELMo achieved to distinguish different meanings of "Sol" in different contexts. The similarity scores on context 1 and context 2 columns are lower than the control column. However, for BERT-500K it can be seen that control context has the lowest similarity although it is expected otherwise. One of the possible reason of this would be that BERT-500K did not finish its training, because when it is continued to get trained, as shown in BERT-6M, it learns to distinguish. The only difference between 6M and 500k versions is the training steps.

In Table 5.5, it can be seen that all models achieved to assign similar vector to similar meanings of the word and dissimilar to different one. Here again the trend of BERT-500K and BERT-ELMo is similar and BERT-6M is better at distinguishing different meanings. All in all, this task shows that BERT-6M is robust in contextual word representations and BERT-ELMo has similar trends with BERT-500K which shows that 6 million training steps for BERT-ELMo could result in same maybe better results with BERT-6M. Additionally, these results show that BERT-ELMo is able to do what BERT can do in terms of contextual word embeddings.

5.1.1.3. Morphological Analysis Results

The evaluation method is same as analogy task's method, however there are additional context to carry hidden information inside suffixes. For example, to show the effect of future tense "-ecek" the definition of word "yarın" needed so that if the word with suffix "-ecek" carries same information with word "yarın". All the contexts for the words can be found in Appendix D. The vector operations and expected results are shown below.

For Negativity:

$V(query) = V(yapildi) - V(yapilmadi) + V(olmadi)$
It is expected that $V(query)$ vector is similar mostly with $V(oldu)$.

For Belonging:
$V(query) = V(kalemlerin) - V(sen) + V(ben)$
It is expected that $V(query)$ vector is similar mostly with $V(kalemlerim)$.

For Tense:
$V(query) = V(geldi) - V(dun) + V(yarin)$
It is expected that $V(query)$ vector is similar mostly with $V(gelecek)$.

In Table 5.6, the similarity results of $query$ vector with the vector of the expected words ("oldu" for negativity, "kalemlerim" for belonging, "gelecek" for tense) are shown only. As it can be seen from the table, the P-score of all vectors are below 0.05 which shows that all the similarities are significantly distinguishable (no close similarity to those words). In morphology task, models are expected to learn the morphological features for Turkish. Although there are many morphological suffix groups, only negativity, belonging and tense suffixes are selected for the task.

Table 5.6. Morphology Test Results.

	Negativity		**Belonging**		**Tense**	
	Sim.	P-score	Sim.	P-score	Sim.	P-score
BERT-6M	0.833	$1\text{x}e^{-178}$	0.437	$9\text{x}e^{-116}$	0.395	0.0
BERT-500K	0.821	$1\text{x}e^{-143}$	0.341	$1\text{x}e^{-53}$	0.369	$1\text{x}e^{-59}$
BERT-ELMo	0.818	$2\text{x}e^{-83}$	0.064	$1\text{x}e^{-9}$	0.396	$1\text{x}e^{-252}$

From the similarity results, it can be interpreted that all the models are close to each other and managed to learn morphological features more or less, except BERT-ELMo's belonging task result. Interestingly, except that cell, every similarity score is close to each other. Before talking about the BERT-ELMo's belonging subtask result, one should mention that BERT-500K and BERT-6M managed to show that they learn these three morphological features well. However, in BERT-ELMo's belonging case, first how

other models perceive these words should be explained again. BERT-6M and BERT-500K divide the word "kalemlerim" into "kalem" and "lerim". For the word vector of whole word, the average of vectors of "kalem" and "lerim" tokens are considered. So BERT-* models have power of intensive morphological analysis of Turkish from two studies that are mentioned in Chapter 4.3. Thanks to this pre-analysis, the feature difference between "kalemlerim" and "kalemlerin" only comes the suffix as it should be. For BERT-ELMo, signals are coming from in different styles. There are kernel size of 1 character length, 2 character length to 7 character length. Each of these throws signals to create initial word embedding for encoder part of the BERT-ELMo. All the substring combination of "kalemlerim" in size 1 to 7 throws features. So 49 different feature vectors are created (discarding the number of kernels for each kernel width) and their importance should be adjusted and this adjustment should be learned for BERT-ELMo to get the morphological features. The number of parameters high and the task is not simple so 500k training steps is very low for this task. It managed to learn features from other tasks (negativity and tense) because they are relatively longer than belonging suffix. For belonging, BERT-ELMo should focus on only 1 feature source out of 49 and it is hard to learn when the number of samples are insufficient. The results on negativity and tense shows that BERT-ELMo has high potential to learn morphological features on its own.

5.1.1.4. Out of Vocabulary Word Evaluation

In the out of vocabulary (OOV) word evaluation three target words are selected and two contexts are collected for each. For each context there is a wug-transformed (replacing target word with word "wug") context as well. Context 1 is selected as target and other one selected as compared context. In Table 5.7, it can be seen that original target word from context 1 is compared with "wug" from wug-transformed context 1, the original target word from context 2 and "wug" from wug-transformed context 2. These contexts can be found in Appendix D. The test is conducted only on BERT-ELMo, because original BERT models throw unknown label ([UNK]) for any OOV word. However BERT-ELMo is able to perceive and create embedding for OOV words and even the tar-

get words ("barış", "savaş", "dünya") are OOV for BERT-ELMo. So even comparison of original target words between context 1 and context 2 is an OOV test.

Table 5.7. Similarity Scores of OOV Words between Original Word in Context 1.

Compared Word	**Barış**	**Savaş**	**Dünya**
Context 1 Wug	1.000	0.168	0.089
Context 2 Original Word	0.783	0.651	0.748
Context 2 Wug	0.783	0.026	−0.040

From the Table 5.7, it can be seen that BERT-ELMo achieved highly good result in original word's comparison between context 1 and context 2. To clarify better, for word "Barış" the rows mean "barış" -> "Wug" from context 1, "Barış" -> "Barış" from context 2 and "Barış" -> "Wug" from context 2. The model achieved high similarities in original target word comparisons and for word "barış" it achieved high similarity for "wug" transformed as well. These high similarity score on original word comparisons indicates that BERT-ELMo is able to assign consistent word embeddings for OOV words.

Table 5.8. Similarity Scores of OOV Words between "Wug" in Context 1.

Compared Word	**Barış**	**Savaş**	**Dünya**
Context 1 Original Word	1.000	0.168	0.089
Context 2 Original Word	0.783	0.031	0.007
Context 2 Wug	0.783	0.733	0.791

However, when the target words are replaced with "wug"s, results dropped dramatically. The "wug" test, actually tells us the ability of capturing the meaning of a word. Because, even though word changes, the context is exactly same, what is told about "wug" or "barış" is exactly same. The dramatic decrease could be explained by the character level perception of BERT-ELMo. Because between "Context 2 Original Word" and "Context 2 Wug" rows of Table 5.7, only "wug" is changed in target word and rest is same. So it can be claimed that BERT-ELMo is sensitive in character-level changes and assigns word level meaning with respect to the characters. This is expected feature of the BERT-ELMo and if we combine this results with results from Table 5.8, one can say that BERT-ELMo assigns vectors well. Because Table 5.8 shows the similarity results of word "wug" from

context 1 between original words from context 1 and 2 and "wug" from context 2. In this table, similarity between original word of context 2 is still low but when target words are replaced with "wug" (the last row), similarity results went higher. The behaviours in Table 5.7 and Table 5.8 shows that BERT-ELMo is sensitive in character changes, but it work well in assigning word embeddings consistently. The character change sensitivity is a feature of BERT-ELMo because it learned a formal and structured source so misspelling was not a case in most of the scenarios. Also one can assume that all word inputs are written correctly, in such case BERT-ELMo is able to assign good embeddings for OOV words as well as known words. Additionally, we believe that if some noisy and informal data source would added to the training set, such as Twitter, BERT-ELMo would learn to be robust in misspelling and be more robust in character changes.

With further analysis on the possible reason of "wug"s being dissimilar, it is considered that a rare character "w" might effect the results. Since the dataset is in Turkish, character "w" is seen very rare so another transformation is tested to see if "w" effected the "wug" results. The new transformation is "vug" where there is no rare-in-turkish characters. The results are shown in Table 5.9 and Table 5.10.

Table 5.9. Similarity Scores of OOV Words between Original Word in Context 1.

Compared Word	**Barış**	**Savaş**	**Dünya**
Context 1 Vug	0.999	0.999	0.999
Context 2 Original Word	0.925	0.760	0.478
Context 2 Vug	0.925	0.760	0.478

Table 5.10. Similarity Scores of OOV Words between "Vug" in Context 1.

Compared Word	**Barış**	**Savaş**	**Dünya**
Context 1 Original Word	0.999	0.999	0.999
Context 2 Original Word	0.925	0.760	0.475
Context 2 Vug	0.925	0.760	0.475

In Table 5.9 and Table 5.10, it can be seen that changing "w" with "v" resulted in better similarity. The similarity between target word and vug-transform of it in same context raised 0.999 similarity. This high similarity between original word and vug-transform

of it also resulted in same similarity trend with other compared word. So it can be said that in "vug", model is able to assign exactly same word vector without any character effect. This tells that model is sensitive in characters of the words because in "wug" it raised highly dissimilarity. Also model has a trade-off for whether to focus more on characters or whole word. One interesting thing is that similarity score between original words are increased for "barış" and "savaş", and decreased for "dünya" in Table 5.9 and Table 5.10. The difference is high enough to say that this phenomena can not be explained by floating point or different random number generator because these vectors are gathered from a pretrained model. No reason is found to explain especially similarity difference of "dünya".

In addition to the consistent assignment of word vectors, it should be questioned that if BERT-ELMo assigned correct word embeddings, because it would be consistently wrong as well. To control this, the target word from context 1 is taken and the cosine similarity of it with respect to the every word from context 2 is calculated. Top 10 most similar words are listed in Figure 5.1. As it can be seen from Figure 5.1, the most similar words are meaningful for human eye. For word "Barış" there are some interestingly high results like "karşılanabilir", "gösterilebilir" and "wwgg". Apart from these unexpected samples, the correct ones are assigned well, so we can say recall is high for this model. The reason of the counterintuitive high results in "Barış" is unclear since it created high similarity both for "karşılanabilir" and "wwgg" where these two words does not even share any character. Only explanation would be underfit of the model. For the other two target words, the results are very intuitive and match with human expectations. So we can say that BERT-ELMo assigns consistently meaningful embeddings for OOV words in a certain level. Also it is better to mention that all models are able to distinguish Turkish characters, however due to an encoding problem on system output, some Turkish specific characters are transformed into their latin correspoings.

For word "Barış"	For word "Savaş"	For word "Dünya"
('barıs', 0.941)	('savas', 0.690)	('dunya', 0.748)
('barıs', 0.783)	('savas', 0.651)	('dunya', 0.733)
('wwgg', 0.767)	('soykırım', 0.397)	('teoride', 0.316)
('barısık', 0.698)	('devlet', 0.292)	('yıldızlar', 0.268)
('karsılanabilir', 0.64)	('politik', 0.281)	('yıldızı', 0.264)
('gosterilebilir', 0.614)	('iktidarların', 0.27)	('gezegenler', 0.262)
('geldiniz', 0.172)	('insan', 0.215)	('turu', 0.240)
('durum', 0.072)	('karsı', 0.201)	('gezegenler', 0.224)
('huzur', 0.072)	('hukuk', 0.197)	('yıldızın', 0.216)
(',', 0.071)	('zira', 0.190)	('yapılar', 0.191)

Figure 5.1. Top 10 Most Similar Words in OOV Task for BERT-ELMo.

5.1.2. Quantitative Analyses

There are two accuracy results. One is Exact Match (EM), the other one is Normalized (Norm.). Exact match is the textual exact match of ground truth answer and the predicted answers. This normalization is needed to show the results clearly. Because in original SQuAD dataset where the Turkish QA dataset is originated its format from, has number of different answer types. For example, In such question "Ozan kaç yılında doğmuştur?", all the answers of "1994", "1994 yılında", "1994'te" are correct actually. So authors of the dataset created bunch of acceptable answers which holds the meaning perfectly as well. In Turkish QA dataset this feature is discarded and there is only one answer. Also that answer is not taken exactly from context. Like when the context mentions the Ozan's birthday as "1994'te", dataset sometimes call that "1994" is the correct answer. So in this case exact match labels it as false prediction. To normalize this behaviour, normalization on answers are done by hand. Some of these normalized samples are given in Appendix C. These samples actually reflect the normalization policy well. Also the ratio of increase between exact match and normalized accuracy is close to each other which indicates that same policy applied when normalizing. The reason that the dataset requires this normalization is actually the set is not created by experts. Dataset is created in a competition project by some graduate students. This study is the first one to use this dataset apart from its creators.

Table 5.11. Question Answering Task Scores.

	EM Accuracy	**Norm. Accuracy**
BERT-6M-5	0.349	0.508
BERT-6M-10	0.394	0.572
BERT-500K-5	0.235	0.378
BERT-500K-10	0.306	0.404
BERT-ELMo-5	0.257	0.403
BERT-ELMo-10	0.344	0.476

Each model is trained in 5 epoch and 10 epoch. Since the training times are very hyper parameter tuning did not done. The epoch is selected as 10 at first, then 5 epoch is selected to see that if 10 was a overfit (if there are more accuracy we can get when we lower the number of epochs, due to overfit). Eventually, all the models increased the accuracy as the epoch increased as can be seen from Table 5.11. The answers for whether 10 is the right number or the selected learning rate is the right one requires hyperparameter tuning.

From results it can be seen that all models increased their accuracy as the epoch increased. BERT-6M reached the best results and even 5 epoch version of it outperformed all models. BERT-ELMo outperformed BERT-500K in each epoch trial and even epoch 5 version of BERT-ELMo got nearly same result as epoch 10 result of BERT-500K. These results show that for QA task the pretrained model's training accuracy is important because BERT-6M outperformed with high margin. But the difference of the scores between BERT-500K and BERT-ELMo can not be explained by pretraining accuracy because BERT-ELMo is trained in same number of steps and the MLM accuracy was lower than BERT-500K. Surprisingly, BERT-ELMo managed to surpassed the result of BERT-500K. This can be reasoned by the OOV word processing power of the BERT-ELMo. Because it is seen that BERT models raised unknown label frequently. This might result in poor signals, since unknown label only gives constant noisy output. BERT-ELMo's capability of processing OOV words let it outperform and even get closer to fully learned (BERT-6M) model. There is not much difference between BERT-ELMo and BERT models other than character level embeddings.

These results indicate that the OOV word processing ability of BERT-ELMo gives

it advantage in downstream tasks. Also even with lower MLM accuracy, it manages to surpass bare BERT model.

5.2. Discussion

BERT model requires morphological pre-analysis for best usage of BERT. This requirement create an overhead because it is already slow in training and requires very powerful GPU's and high memory. These requirements make it hard to use, especially for underrepresented languages. The least studied languages in NLP have disadvantage of morphological analysis in their natural language. BERT-ELMo solves this problem with its character level perception because there is no morphology or language for it. If characters are defined, it learns the language and syntactic structures over characters. With this big improvement, if BERT-ELMo model keep the abilities of BERT in it, then it would be a clear improvement without losing anything.

From the results on both qualitative and quantitative, it can be seen that BERT-ELMo has almost the same capability of BERT in word level semantics. There might be unexpected results on some specific examples but all in all the similarity of the behaviour is in an acceptable level to say that BERT-ELMo has same capabilities with BERT. There are also things to say about the pretraining of BERT and BERT-based (such as BERT-ELMo) models. BERT or Transformer structure is sensitive with lots of parameters. There are studies to show that learning rate or initial weight distribution leads BERT into different performances which are surprisingly different. So this sensitivity could lead to questioning about the training. The hyperparameter tuning makes BERT results more robust since this sensitivity is already known. Due to inefficient architecture and long data/training step need of BERT, hyperparameter tuning is often discarded.

For downstream tasks in common practice more than one task is used. However, in Turkish there was not any publicly available datasets when these analyses are conducted for this study. Only QA task is tested because the dataset was available for us. It was not publicly available as well, so there is still need for robust, well-created, carefully designed downstream task datasets in Turkish.

There are much more parameters in BERT-ELMo than BERT. The CNN part has kernel weights to learn which are specified in Chapter 4.2.2. Also highway networks to turn CNN outputs into 1-D embedding vectors has several parameters as well. To adjust both BERT parameters and character level parameters is harder task and requires much more data and training steps than plain BERT. Even though there is longer training requirement for BERT-ELMo, it achieved well results when compared with BERT models. So character-level embedding layer improved more than lower training of BERT-ELMo creates disadvantage.

Last thing to mention that, in order to clarify the morphological structure learning ability of BERT-ELMo, more languages are needed as subject. Turkish might be the sweet point for BERT-ELMo or it might fail in morphologically less complex languages. The robustness of the model across languages is also another metric to measure for this kind of study. Due to long training times, it was not possible to train models in another language.

All in all, BERT-ELMo showed similar results with BERT in most of the tests and even surpassed in some tasks. This showed that BERT-ELMo is robust in capturing semantics. Thanks to its character-level part, BERT-ELMo can be used in any language without any pre-analysis. Adding this feature to BERT is very important because BERT requires intense pre-analysis but BERT-ELMo requires no pre-training and can adjust to any language if the characters are present in vocabulary. One can say that BERT-ELMo keeps the abilities of BERT and adds being language agnostic and OOV word processing over it.

CHAPTER 6

CONCLUSION

Pretrained language models are very popular in natural language processing. A language model is trained with a large dataset with huge repetition, then this model is fine-tuned on downstream tasks such as question answering, named entity recognition, or natural language inference. There are two levels of perception in language modeling. One is word level perception where model perceives words as atomic, the other is character level perception where model learns language character by character. There are pros and cons of both views. Moreover, there are hybrid models where the power of perceiving in character level is combined with that of word level. These models are popular recently and they try to find a good way to combine these approaches.

BERT is a masked language model that uses word level perception with morphological adjustments in vocabulary. Even though there is morphological decomposition, model still gets unknown words and intensive morphological analysis is required before training. This requirement and unknown word problem put an overhead for BERT. On the other hand, ELMo model has a character level perception and produces contextual word embeddings as well. BERT outperforms ELMo in most of the tasks with a high margin when it is fully configured.

In this thesis, BERT language model is improved with an additional embedding layer which is inspired by ELMo. The new proposed model is called as BERT-ELMo, because it is a combination of two successful models. Unlike BERT, which relies on word level perception, proposed model uses character level perception. However, it carries this character level information to word level so that word's meaning is searched through its character information. To the best of our knowledge, this study is the first one that uses Turkish language as a subject in BERT model training with a morphological pre-analysis and first study to embed a character level perception to BERT. BERT-ELMo is expected to do two things, one is preserving the BERT model's ability of creating robust contextual word embeddings or improving them and the other one is processing OOV

words well so that no morphological pre-analysis will be needed and no unknown word label will be raised. To test these abilities, both qualitative and quantitative analyses were conducted and results showed that BERT-ELMo model has the same word level capability as BERT. Moreover, these analyses showed that BERT-ELMo's character level view helps it in being language agnostic and capable to process OOV words so that no information is missed from the source.

As a future work, the model can be trained in different language types with respect to their complexity in morphology. The difference between performance of model in morphologically rich languages and morphologically less complex languages could strengthen the hypothesis. Also proposed model is in under-fit state. The performance in fully-trained state could end up in more robust results. Also the qualitative analyses might be quantified so the morphological analyses would show more robust and better results. Additionally, only question answering downstream task is used in this study. More downstream tasks can strengthen the hypothesis and show the potential or weakness of the proposed model better.

REFERENCES

Ahmet Aksoy (2017). Kalbur. In *https://github.com/ahmetax/kalbur*. Github.

Alec Radford (2018). Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). Language models are unsupervised multitask learners.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Brussels, Belgium, pp. 353–355. Association for Computational Linguistics.

Allen M. Turing (1950). Computing machinery and intelligence.

Andrei Alexandrescu and Katrin Kirchhoff (2006). Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, USA, pp. 1–4. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). Attention is all you need. In *NIPS*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher (2017). Learned in translation: Contextualized word vectors. In *NIPS*, pp. 6297–6308.

Dan Hendrycks and Kevin Gimpel (2016). Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR abs/1606.08415*.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf (2016). Semi-supervised word sense disambiguation with neural models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan, pp. 1374–1385. The COLING 2016 Organizing Committee.

Dongyun Liang, Weiran Xu, and Yinge Zhao (2017). Combining word-level and character-level representations for relation classification of informal text. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, Vancouver, Canada, pp. 43–47. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2014). Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Frank Rosenblatt (1957). The perceptron: A perceiving and recognizing automaton (project para). report no. In *Cornell Aeronautical Laboratory*, pp. 85–460–1.

Gabriella Vigliocco, Lotte Meteyard , Mark Andrews, and Stavroula Kousta (2009). Toward a theory of semantic representation. *Language and Cognition 1*(2), 219–247.

Geoffrey E. Hinton, James L. McClelland, and David E. Rumelhart (1986). *Distributed Representations*, pp. 77–109. Cambridge, MA, USA: MIT Press.

George A. Miller (1995). Wordnet: A lexical database for english. Volume 38, pp. 39–41. Communications of the ACM.

Henry J. Kelley (1960). Gradient theory of optimal flight paths. *Ars Journal 30*(10), 947–954.

Hinrich Schütze (1993). Word space. In S. J. Hanson, J. D. Cowan, and C. L. Giles (Eds.), *Advances in Neural Information Processing Systems 5*, pp. 895–902. Morgan-Kaufmann.

Huadong Chen, Shujian Huang, David Chiang, Xinyu Dai, and Jiajun Chen (2018). Combining character and word information in neural machine translation using a multi-level attention. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana, pp. 1284–1293. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Jean Berko (1958). The child's learning of english morphology. *WORD 14*(2-3), 150–177.

Jeffrey L. Elman (1990). Finding structure in time. Volume 14, pp. 179–211.

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

John McCarthy (1960). Recursive functions of symbolic expressions and their computation by machine, part i. *Communications of the ACM*.

John R. Firth (1957). A synopsis of linguistic theory 1930-55. In *Studies in Linguistic Analysis (special volume of the Philological Society)*, Volume 1952-59, pp. 1–32. The Philological Society.

Kenneth C. Litkowski (2002). Sense information for disambiguation: Confluence of supervised and unsupervised methods. In *Proceedings of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, pp. 47–53. Association for Computational Linguistics.

Kim Yoon, Jernite Yacine, Sontag David, and Alexander M. Rush (2016).

Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 2741–2749. AAAI Press.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana, pp. 2227–2237. Association for Computational Linguistics.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power (2017). Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1756–1765. Association for Computational Linguistics.

Mikael Kågebäck and Hans Salomonsson (2016). Word sense disambiguation using a bidirectional LSTM. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, Osaka, Japan, pp. 51–56. The COLING 2016 Organizing Committee.

Minh Le, Marten Postma, Jacopo Urbani, and Piek Vossen (2018). A deep dive into word sense disambiguation with LSTM. In *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA, pp. 354–365. Association for Computational Linguistics.

Noam Chomsky (1957). Syntactic structures.

Orhan Bilgin (2016). Biçimbilimsel bakımdan karmaşık türkçe kelimelerin İşlenmesinde frekans etkileri. In *(yayınlanmamış yüksek lisans tezi)*, İstanbul. Boğaziçi Üniversitesi.

Peter D. Turney, Michael L. Littman, Jeffrey Bigham, and Victor Shnayder (2003). Combining independent modules to solve multiple-choice synonym and analogy problems. *CoRR cs.CL/0309035*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov (2016). Enriching word vectors with subword information. *ACL 5*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, pp. 2383–2392. Association for Computational Linguistics.

Rada Mihalcea and Andras Csomai (2005). SenseLearner: Word sense disambiguation for all words in unrestricted text. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, Ann Arbor, Michigan, pp. 53–56. Association for Computational Linguistics.

Rada Mihalcea and Ehsanul Faruque (2004). SenseLearner: Minimally supervised word sense disambiguation for all words in open text. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, pp. 155–158. Association for Computational Linguistics.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu (2016). Exploring the limits of language modeling. *ArXiv abs/1602.02410*.

Rico Sennrich, Barry Haddow, and Alexandra Birch (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, pp. 1715–1725. Association for Computational Linguistics.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber (2015). Highway networks. *CoRR abs/1505.00387*.

Sarthak Jain and Byron C. Wallace (2019). Attention is not explanation. In *NAACL-HLT*.

Sepp Hochreiter (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 6*(2), 107–116.

Sepp Hochreiter and Jürgen Schmidhuber (1997). Long short-term memory. Volume 9, Cambridge, MA, USA, pp. 1735–1780. MIT Press.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher (2018). Regularizing and optimizing lstm language models. *ArXiv abs/1708.02182*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, Red Hook, NY, USA, pp. 3111–3119. Curran Associates Inc.

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

Yann Lecun and Yoshua Bengio (1995). Convolutional networks for images, speech, and time-series. In M. Arbib (Ed.), *The handbook of brain theory and neural networks*. MIT Press.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov (2019). Roberta: A robustly optimized bert pretraining approach.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian,

Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR abs/1609.08144*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin (2003). A neural probabilistic language model. *J. Mach. Learn. Res. 3*, 1137–1155.

Zellig S. Harris (1954). Distributional structure.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut (2019). Albert: A lite bert for self-supervised learning of language representations.

APPENDIX A

QUESTION ANSWERING TRAINING LOSS GRAPHS

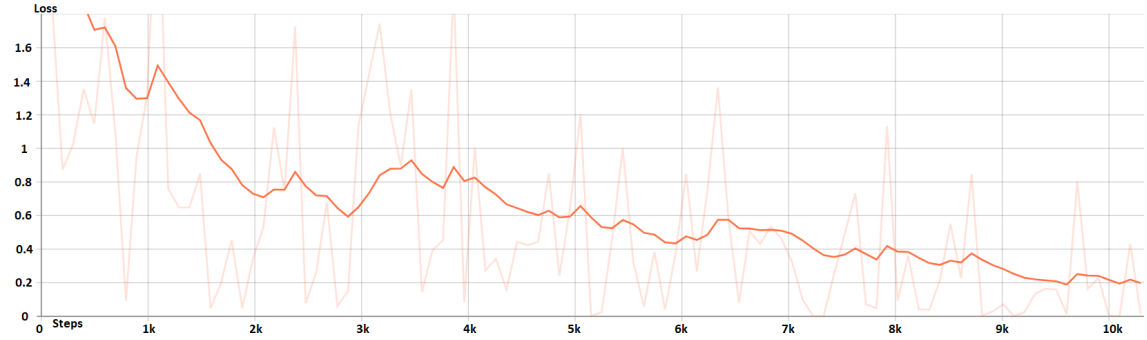

Figure A.1. Loss Graph of BERT-6M in Question Answering Task for 10 epochs.

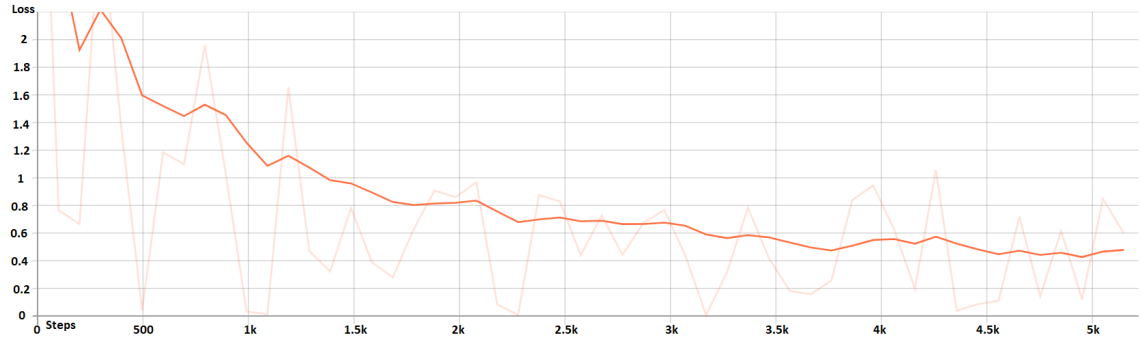

Figure A.2. Loss Graph of BERT-6M in Question Answering Task for 5 epochs.

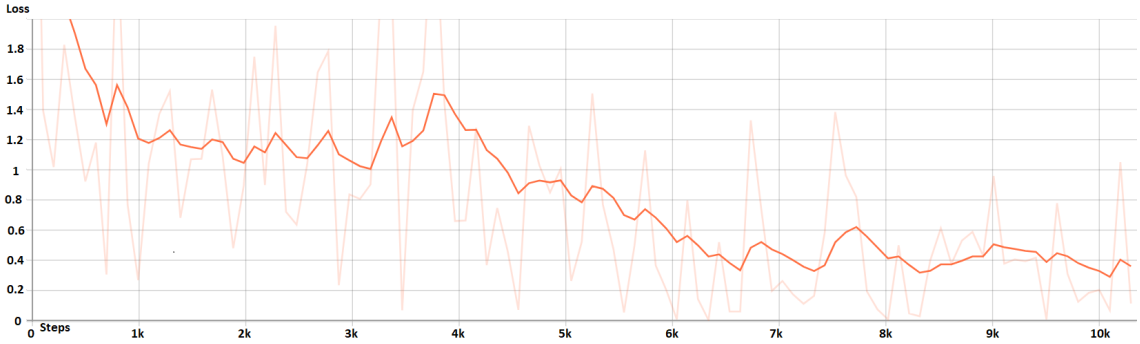

Figure A.3. Loss Graph of BERT-500k in Question Answering Task for 10 epochs.

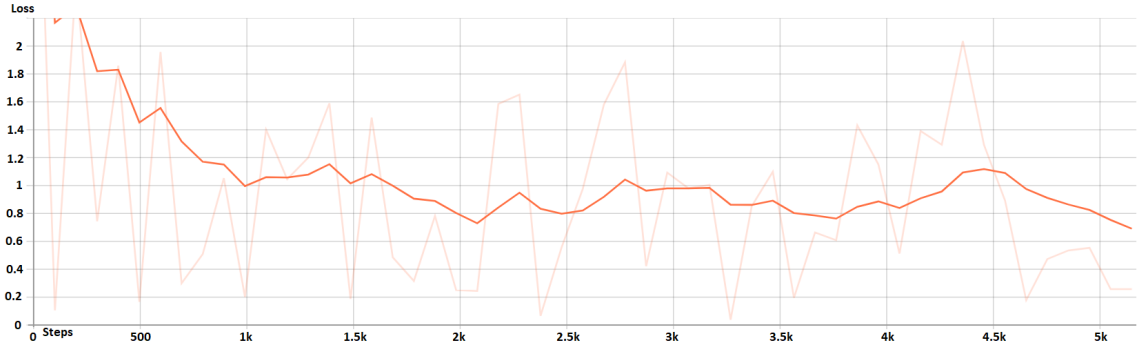

Figure A.4. Loss Graph of BERT-500k in Question Answering Task for 5 epochs.

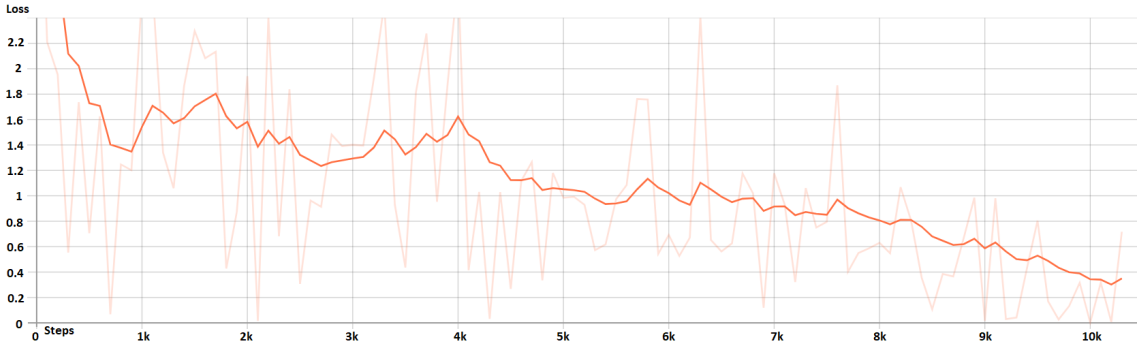

Figure A.5. Loss Graph of BERT-ELMo in Question Answering Task for 10 epochs.

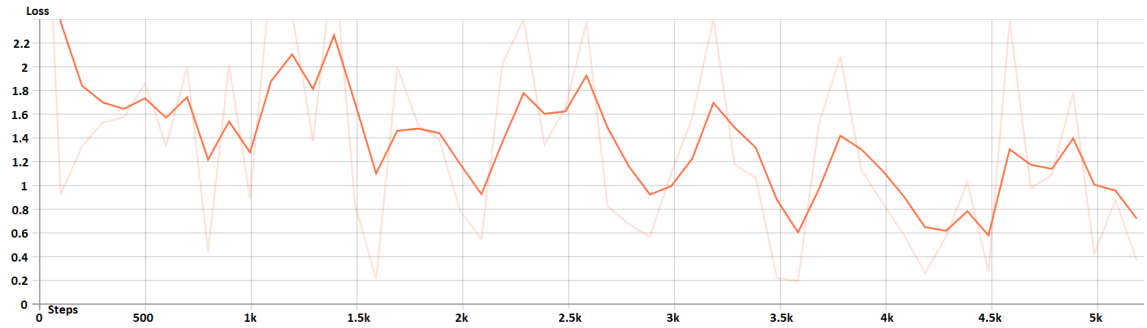

Figure A.6. Loss Graph of BERT-ELMo in Question Answering Task for 5 epochs.

APPENDIX B

LANGUAGE MODELING TRAINING LOSS GRAPHS

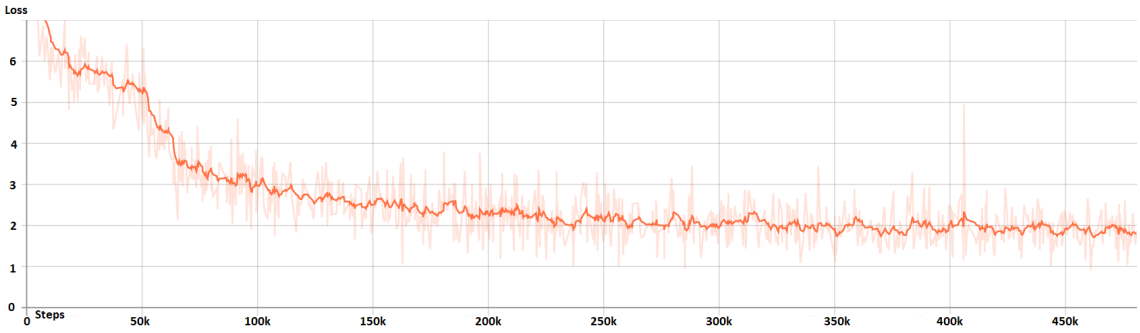

Figure B.1. Loss Graph of BERT-500k Language Model Training.

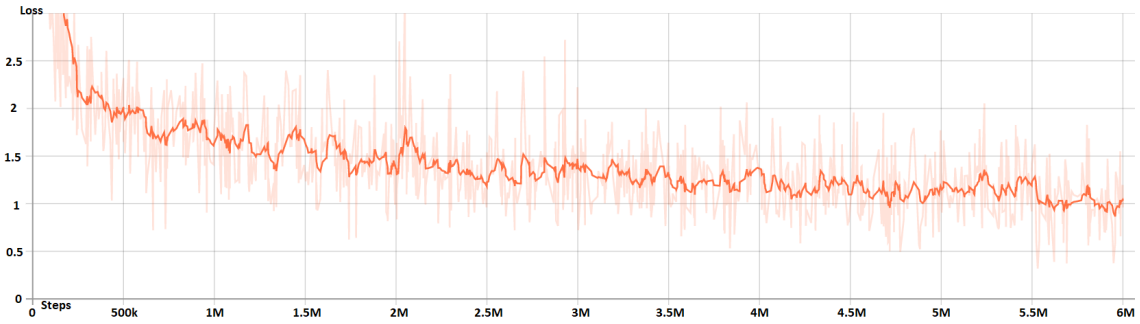

Figure B.2. Loss Graph of BERT-6M Language Model Training.

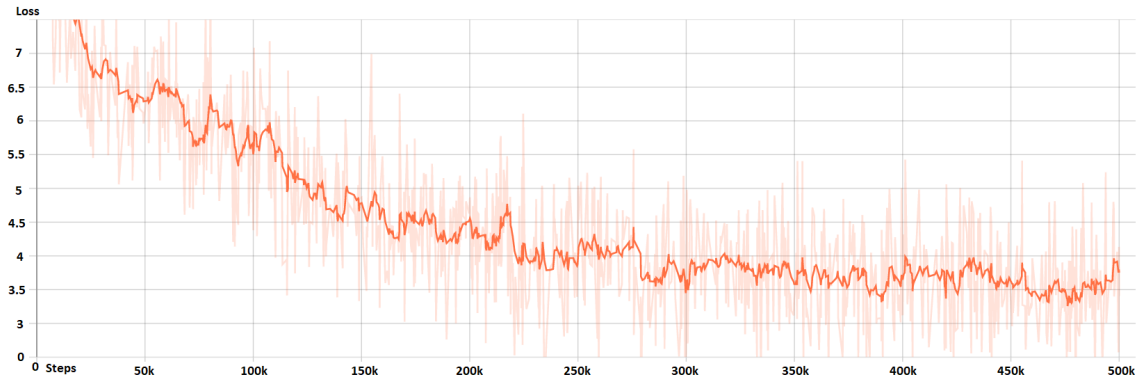

Figure B.3. Loss Graph of BERT-ELMo Language Model Training.

APPENDIX C

SAMPLES OF NORMALIZED PREDICTIONS

These samples are identified as false prediction in exact match calculation. Due to the correctness of the predicted answers in terms of meaning, they are accepted as true in normalized accuracy calculation. These are not the only ones but just some sample of them gathered from BERT-ELMo Epoch-10 predictions. Also these questions have context which includes the answer span, however they vary in text size, that is why context are not included.

Question: Kemaleddin ibn Yunus ilk eğitimini kimin yanında almıştır?
True Answer: Şeyh Yunus Rızauddin
Predicted Answer: Şeyh Yunus Rızauddin'in yanında

Question: Kemaleddin ibn Yunus, Bağdat'ta nerede okumaya devam etmiştir?
True Answer: Nizamiye Medreseleri
Predicted Answer: Nizamiye Medreseleri'nde

Question: Pardus'un ilk kurulabilir sürümü hangi tarihte ağ üzerinden paylaşılmıştır?
True Answer: 27 Aralık 2005'te
Predicted Answer: 27 Aralık 2005

Question: 19 Haziran 2012'de Pardus projesinin toplantısında Pardus'un hangi projedeki akıllı tahtalarda kulanılacağı açıklanmıştır?
True Answer: FATİH
Predicted Answer: FATİH Projesindeki

Question: Havâdisü'd-Duhûr fî Mede'l-Eyyâm ve'ş-Şuhûr adlı eserin İstanbul'daki yazması hangi kütüphanededir?

True Answer: Ayasofya Kütüpahnesi'ndedir
Predicted Answer: Ayasofya Kütüpahnesi

Question: Akdeniz Üniversitesi ne zaman kurulmuştur?
True Answer: 20 Temmuz 1982'de
Predicted Answer: 20 Temmuz 1982

Question: Feza Günergun hangi tarihte yardımcı doçent olmuştur?
True Answer: 1987
Predicted Answer: 1987 yılında

Question: Gözlemevi hangi yıllar arasında inşa edilmiştir?
True Answer: 1934-1936
Predicted Answer: 1934-1936 yılları arasında

Question: Süreyya Ciliv 2009 yılında hangi ödülü almıştır?
True Answer: World Communication Awards'ta 2009 yılının CEO'su ödülü
Predicted Answer: World Communication Awards'ta 2009 yılının CEO'su ödülünü

Question: Taktik İnsansız Hava Aracı Sistemi Geliştirme programı testleri sırasında hangi özellikler test edilmiştir?
True Answer: tam otomatik taksi, kalkış, uçuş, iniş, frenleme ve tekrar hangara dönüş gibi özellikler
Predicted Answer: tam otomatik taksi, kalkış, uçuş, iniş, frenleme ve tekrar hangara dönüş

Question: Savunma Sanayi İcra Kurul'unda kaç sene sonra sözleşme olmuştur?
True Answer: 2 yıl sonra
Predicted Answer: 2 yıl

Question: Şükrullâh'ın yazdığı Behcetü't Tevârîh isimli eser hangi Osmanlı tarihçileri

tarafından kaynak olarak kullanıldı?

True Answer: özellikle Karamani Mehmet Paşa, Sarıca Kemal, Ruhi Çelebi, Mehmet Zaim, kaynak olarak kullanılmıştır

Predicted Answer: Karamani Mehmet Paşa, Sarıca Kemal, Ruhi Çelebi, Mehmet Zaim

APPENDIX D

CONTEXT PARAGRAPHS FOR QUALITATIVE ANALYSES

Paragraphs that are used in analogy analysis section:

Gender Analogy:

Kral, belirli bir ulus ya da bölge üzerinde egemen olan hükümdar. İmparatordan sonraki en yüksek seküler hükümdarlık makamıdır. Dünyanın pek çok bölgesinde karşılaşılan krallık çoğunlukla babadan çocuklara geçer. Bununla birlikte Ortaçağ Almanya'sındaki gibi seçimle başa gelen krallar da mevcuttur. Genellikle çoğu kral erkektir. ||| *wwgg*

Kraliçe, Batı uygarlığında halkı yöneten tek güç ve otorite olan kadın yönetici, monark. Kraliçe, iktidarının kaynağını kral hayatta ise, ondan alır. Bunun yanı sıra kraliçe unvanını ebeveynlerinden alan ve eşi kral olmayan kraliçeler de mevcuttur. Kraliçe unvanını ebeveynlerinden alan ve eşi kral olmayan kraliçeyi diğerlerinden ayırmak için "hükümran kraliçe" (Fr. reine regnante) tabiri kullanılır. Hükümran kraliçelerin eşleri kral değildir ve prens konsor ya da sadece prens olarak adlandırılırlar. ||| *wwgg*

Prens, Avrupa hanedanlarında kral olmayan erkek üyelere verilen genel soyluluk unvanıdır. Osmanlı'daki şehzade unvanına denk gelir. Prensler çoğu ülkede prenseslerle birlikte taht için sıraya girerlerdi. Fransa'da veliaht prensine döfen denirdi. Eski Türklerde Tigin sözcüğü prens sözcüğüne karşılık gelir. ||| *wwgg*

Prenses, genellikle Avrupa hanedanlarında kadın üyelere verilen genel soyluluk unvanı. Bazı ülkelerde tahta çıkma şansı yokken bazı ülkelerde tek başlarına tahta çıkabilirler. Tarihte bağlı bulundukları ülke ve hanedanların çıkarları için diğer ülkelerin hükümdarları veya önemli görevlileriyle evlendirilmişlerdir. Örnek olarak sıkça görülen Bizans hanedanıyla ve Osmanlı hanedanları arasındaki evliliklerdir. Prenses sözcüğünün Türkçe karşılığı Kunçuy dur. ||| *wwgg*

Capital Analogy:

Türkiye'nin başkenti Ankara'dır. Ülkenin en büyük idari birimleri illerdir ve 81 il vardır. Bu iller ilçelere ayrılmıştır, toplamda 973 ilçe mevcuttur. ||| *wwgg*

Fransa'da ayrıca uzunluğu toplamda 893.300 kilometreyi bulan bir karayolu ağı da bulunmaktadır. Başkent Paris ve çevresi en yoğun yol ve otoyol ağıyla örülmüş durumdadır. ||| *wwgg*

Plural Analogy:

Neptün'ün halkaları, Neptün etrafında yer alan ve beş ana halkadan oluşan bir halka sistemidir. Başta "yaylar" olarak adlandırılan halkalar, 22 Temmuz 1984'te Patrice Bouchet, Reinhold Häfner ve Jean Manfroid'dan oluşan ekip tarafından Şili'deki La Silla Gözlemevi'nde ve William Hubbard liderliğindeki bir program kapsamında F. Vilas ve L. R. Elicer tarafından Cerro Tololo Amerikaarası Gözlemevi'nde keşfedildi. ||| *wwgg*

Kurşun kalem, kâğıt üzerine yazı veya çizim için kullanılan, yazıcı kısmı çoğunlukla kil ve grafitten üretilen kalem. Tipik bir kurşun kalemde grafitin etrafı ahşap kaplıdır. Bunun yanı sıra metal veya plastik muhafazaya sahip kurşun kalemler de mevcuttur. ||| *wwgg*

Paragraphs that are used in contextuality analysis section:

For the word "Dil":

The paragraph of the target context for "Dil":

Dil veya lîsan, insanlar arasında anlaşmayı sağlayan doğal bir araç, kendisine özgü kuralları olan ve ancak bu kurallar içerisinde gelişen canlı bir varlık, temeli tarihin bilinmeyen dönemlerinde atılmış bir gizli anlaşmalar düzeni, seslerden örülmüş toplumsal bir kurumdur. ||| *wwgg*

First paragraph from compared context for "Dil":

Dil, ağız içinde bulunan ve tat alma duyusunu gerçekleştiren, kaslardan yapılmış bir organdır. Ayrıca yiyecekleri çiğneme ve yutma işlemlerine yardım eder, insanlarda konuşmayı da sağlar. ||| *wwgg*

Second paragraph from compared context for "Dil":

Dile pürüzlü bir görünüm veren,dilin üst yüzeyinde ve yanlarında yer alan minik çıkıntılara verilen isimdir. İçlerinde tat tomurcukları bulundururlar. Bu tomurcuklar içerisinde ise tat hücreleri vardır. İnsan dilinin her yeri farklı tatları hisseder. Dil ucu "tatlı" , ucun hemen arkası "tuzlu", dilin yanları "ekşi" ve arkası "acı" tatlarını hisseden algılayıcılar barındırır. ||| *wwgg*

The paragraph of the control context for "Dil":

Dil, kuşaklar arasında ve aktüel durumda insanlığın kullandığı bağdır. Bu bağ kültürün taşıyıcısıdır. Bundan dolayıdır ki, dil ve kültür birbirini sürekli etkileyen iki olgudur. Bu iki olgudan herhangi birinde olan değişiklik diğerini de etkiler. Bu da doğal bir süreklilik ve tabii olma durumunu doğurur. Dil, toplumda var olan bir gerçekliktir. ||| *wwgg*

For the word "Sol":

The paragraph of the target context for "Sol":

Mesela Birleşik Krallık'taki İşçi Partisi -Birleşik Krallık'ta sol bu partiyle özdeşleşmiştir- çoğunlukla küresel kapitalizmi savunmaktadır ve dünyadaki en kapitalist ekonomilerden birini meydana getirmiştir. Diğer bir uç örnek ise Zimbabve'de sol ile özdeşleştirilen Zimbabve Afrika Ulusal Birliği-Vatanseverler Cephesi'dir; dünyadaki en sosyalist ekonomilerden birini gerçekleştirmiştir. ||| *wwgg*

First paragraph from compared context for "Sol":

Solaklık (sinistralite olarak da bilinir), sağ el yerine sol eli kullanmak, sola yatkınlık. Solaklar günlük aktivitelerde özellikle el yazısı ve yazı yazmada sol ellerini kullanırlar. Eski zamanlarda, özellikle Orta Çağ Avrupasında, solaklık kötülüğün, cadılığın ve şeytanın simgesi olarak görülmüş ve solaklar ayrımcılığa maruz bırakılmıştır. ||| *wwgg*

Second paragraph from compared context for "Sol":

Pek çok Müslüman çoğunluğa sahip ülkede sol eli kullanmak, özellikle sol ile yemek yemek veya selamlaşmak, tabu bir davranış olarak görülmekte ve solaklar sağ elin kullanılmasına zorlanmakta veya teşvik edilmektedir. İslam'da haram olmamak ile birlikte bir hadiste geçen sol eli ile yemek yemekte olan bir adama İslam dinince peygamber kabul edilen Muhammed'in sağ eli ile yemek yemesini söylemesi ve bu isteğe uyulmayınca beddua etmesi İslam'ın yaygın olduğu toplumlarda sağ el kullanımına teşviğin başlıca nedenlerinden biridir. Aynı zamanda sol el bu ülkelerde genellikle tuvalette kullanılmakta ve pis görülmektedir. ||| *wwgg*

The paragraph of the control context for "Sol":

Sol siyaset kavramının kökeni Fransız İhtilali dönemine dayanır. İhtilal sonrası kurulan parlamentoda özgürlüklerin destekçisi olan halkçılar genellikle başkan koltuğunun solunda oturmaktaydılar. Değişimlere karşı çıkmakta olan zenginler, burjuva kişiler ise sağda otururlardı. Bugün Fransız parlamentosunda bu gelenek hala devam etmektedir.

||| *wwgg*

For the word "Yüz":

The paragraph of the target context for "Yüz":

Yüz, bir sayı. Doğal sayı sisteminde 101'den önce yer alır ve 99'dan sonra gelir. 10 sayısının karesi, 10000 sayısının kare köküdür. ||| *wwgg*

First paragraph from compared context for "Yüz":

Yüz, canlıların başlarının ön bölümüne verilen isim: alın, gözler, burun, yanaklar, ağız ve çenenin oluşturduğu bütün. Surat, sima, çehre kelimeleri ile eşanlamlıdır. ||| *wwgg*

Second paragraph from compared context for "Yüz":

Yüz insanların kafalarının öne tarafındaki bölümün adıdır. Hissetme hariç bütün his duyularının yerleştiği, insanların birbirini tanıyabilmesi için çok önemli olan bir organdır. ||| *wwgg*

The paragraph of the control context for "Sol":

yüz, doksan dokuz ile yüz bir arasındaki bir doğal sayıdır. Üç basamaklı en küçük doğal sayı olup ikiye, beşe, ona, yirmiye, ve elliye tam bölünür. ||| *wwgg*

Paragraphs that are used in morphological analysis section:

Effect of Negativity Suffix (-me, -ma):

Meclis-i Mebusan üyelerini belirlemek için Ali Rıza Paşa hükûmeti döneminde seçimler yapıldı. Anadolu ve Rumeli Müdafaa-i Hukuk Cemiyeti üyeleri seçimlerde başarılı oldu. ||| *wwgg*

Meclis-i Mebusan üyelerini belirlemek için Ali Rıza Paşa hükûmeti döneminde seçimler yapılmadı. Anadolu ve Rumeli Müdafaa-i Hukuk Cemiyeti üyeleri seçimlerde başarılı olmadı. ||| *wwgg*

Effect of Belonging Suffix (-m, -n):

Kalemlerim birisi tarafından çalındı ama kimin çaldığını bulamadım. Kötü bir niyetle çalınmadığını düşünüyorum ama nasıl yazı yazacağım bilemiyorum. Peki kalemlerin ne halde, sağ salim yerlerinde duruyorlar mı? ||| *wwgg*

Sen, teklik ikinci kişiyi gösteren söz. Varoluşçu felsefede sen, ötekiler içinde bize en yakın olanıdır, ya da bize eşlik eden ötekidir. ||| *wwgg*

Ben (kişi adılı) teklik birinci kişiyi gösteren söz, kişi zamiri. Dervişlik olaydı taç ile hırka, ben de alırdım otuza kırka. ||| *wwgg*

Effect of Tense Suffix (-di, -ecek):

Akşam için kendimizi şımartalım dışarıya çıkalım derken annemler geldi. Bütün akşam bizde durdular ve tek tatilimiz de yok oldu. Önümüzdeki tatillerde umarım yalnız kalabiliriz ||| *wwgg*

Akşam için kendimizi şımartalım dışarıya çıkalım demiştik ama annemler gelecek. Bütün akşam bizde duracaklar ve tek tatilimiz de yok olacak. Önümüzdeki tatillerde umarım yalnız kalabiliriz ||| *wwgg*

Yarın , bugünden sonraki gün. Yarın yaşanacak olaylardan kendimi sorumlu tutmuyorum. ||| *wwgg*

Dün , bugünden önceki gün. Dün yaşanmış olaylardan kendimi sorumlu tutmuyorum. ||| *wwgg*

Paragraphs that are used in out of vocabulary analysis section:

First paragraph for word "Dünya":

Dünya, Yer veya Yerküre, Güneş Sistemi'nde Güneş'e en yakın üçüncü gezegen olup şu an için üzerinde yaşam ve sıvı su barındırdığı bilinen tek astronomik cisimdir. Radyometrik tarihleme ve diğer kanıtlara göre 4,5 milyar yıldan fazla süre önce oluşmuştur. Dünya'nın yerçekimi, uzaydaki diğer nesnelerle, özellikle Güneş'le ve tek doğal uydusu Ay'la etkileşime girer. Dünya'nın Güneş'in etrafındaki yörüngesi, 365,256 güneş gün, yani bir yıldız yılı sürer. ||| *wwgg*

Wug transformation in first paragraph for word "Dünya":

Wug, Yer veya Yerküre, Güneş Sistemi'nde Güneş'e en yakın üçüncü gezegen olup şu an için üzerinde yaşam ve sıvı su barındırdığı bilinen tek astronomik cisimdir. Radyometrik tarihleme ve diğer kanıtlara göre 4,5 milyar yıldan fazla süre önce oluşmuştur. Wug'un yerçekimi, uzaydaki diğer nesnelerle, özellikle Güneş'le ve tek doğal uydusu Ay'la etkileşime girer. Wug'un Güneş'in etrafındaki yörüngesi, 365,256 güneş gün, yani bir yıldız yılı sürer. ||| *wwgg*

Second paragraph for word "Dünya":

Teoride, karmaşık bileşik yapılar ve içerdiği elementler göze alındığında, Güneş, Dünya ve diğer gezegenler dahil Güneş Sistemi'ndeki yapıları oluşturan moleküler bulutsunun kaynağı, ömrünü önceden tamamlamış bir genç tip yıldızın dağılmış artıklarının ve yıldızlar arası maddenin bir merkez etrafında dönerek gittikçe yoğunlaşmasıyla oluşmuştur. Merkezde yoğunlaşan hidrojen ve helyum molekülleri yeni bir G2 türü yıldızı, yani Güneş'i oluşturmaya başlamış, çevre disklerdeki yoğunluklu bölgelerde ise gezegenler oluşmaya başlamıştır. Dünya ise Güneş'e 3. sırada yakınlıkta bulunan karasal bir iç gezegendir. ||| *wwgg*

Wug transformation in second paragraph for word "Dünya":

Teoride, karmaşık bileşik yapılar ve içerdiği elementler göze alındığında, Güneş, Wug ve diğer gezegenler dahil Güneş Sistemi'ndeki yapıları oluşturan moleküler bulutsunun kaynağı, ömrünü önceden tamamlamış bir genç tip yıldızın dağılmış artıklarının ve yıldızlar arası maddenin bir merkez etrafında dönerek gittikçe yoğunlaşmasıyla oluşmuştur. Merkezde yoğunlaşan hidrojen ve helyum molekülleri yeni bir G2 türü yıldızı, yani Güneş'i oluşturmaya başlamış, çevre disklerdeki yoğunluklu bölgelerde ise gezegenler oluşmaya başlamıştır. Wug ise Güneş'e 3. sırada yakınlıkta bulunan karasal bir iç gezegendir. ||| *wwgg*

First paragraph for word "Savaş":

Savaş veya harp; ülkeler, hükûmetler, bloklar ya da bir ülke içerisindeki toplumlar, isyancılar veya milisler gibi büyük gruplar arasında gerçekleşen topyekûn silahlı mücadeledir. Savaşlar genellikle dini, millî, siyasi ve ekonomik amaçlara ulaşmak için gerçekleştirilir. ||| *wwgg*

Wug transformation in second paragraph for word "Savaş":

Wug veya harp; ülkeler, hükûmetler, bloklar ya da bir ülke içerisindeki toplumlar, isyancılar veya milisler gibi büyük gruplar arasında gerçekleşen topyekûn silahlı mücadeledir. Wuglar genellikle dini, millî, siyasi ve ekonomik amaçlara ulaşmak için gerçekleştirilir. ||| *wwgg*

Second paragraph for word "Savaş":

Ancak tüm bu ayrımlar politik birer yargıdır, iktidarların uygulamalarından öte uluslararası hukuk da bu konuda net bir ayrım yapmamakta ve en ciddi insan hakları ihlallerine karşı bile örgütlü bir direniş hakkı tanımlamamaktadır. Zira bu durum soykırım

ve devlet terörü gibi suçlara karşı savaş hakkını engellemekte, insanlığa karşı suçlar ve büyük çapta savaş suçlarına karşı mücadeleyi zayıflatmaktadır. ||| *wwgg*

Wug transformation in second paragraph for word "Savaş":

Ancak tüm bu ayrımlar politik birer yargıdır, iktidarların uygulamalarından öte uluslararası hukuk da bu konuda net bir ayrım yapmamakta ve en ciddi insan hakları ihlallerine karşı bile örgütlü bir direniş hakkı tanımlamamaktadır. Zira bu durum soykırım ve devlet terörü gibi suçlara karşı wug hakkını engellemekte, insanlığa karşı suçlar ve büyük çapta wug suçlarına karşı mücadeleyi zayıflatmaktadır. ||| *wwgg*

First paragraph for word "Barış":

Barış kelimesi genel anlamda düşmanlığın olmaması anlamında kabul görülür. Başka bir anlatımla kötülükten, kavgalardan, savaşlardan kurtuluş, uyum, birlik, bütünlük, sükunet, sessizlik, huzur içinde yaşamak olarak da tanımlanabilir. ||| *wwgg*

Wug transformation in second paragraph for word "Barış":

Wug kelimesi genel anlamda düşmanlığın olmaması anlamında kabul görülür. Başka bir anlatımla kötülükten, kavgalardan, wuglardan kurtuluş, uyum, birlik, bütünlük, sükunet, sessizlik, huzur içinde yaşamak olarak da tanımlanabilir. ||| *wwgg*

Second paragraph for word "Barış":

Barış halk arasında hoş geldiniz olarak da karşılanabilir. Barış kelimesi duygusal bir durum içinde kullanılabilir. Bir insanın kendisiyle barışık olması, kendi içinde bir denge, sakinlik, huzur içinde olması buna örnek gösterilebilir. ||| *wwgg*

Wug transformation in second paragraph for word "Barış":

Wug halk arasında hoş geldiniz olarak da karşılanabilir. Wug kelimesi duygusal bir durum içinde kullanılabilir. Bir insanın kendisiyle wuguk olması, kendi içinde bir denge, sakinlik, huzur içinde olması buna örnek gösterilebilir. ||| *wwgg*