A DEEP LEARNING-BASED EXTRACTIVE TEXT SUMMARIZATION SYSTEM FOR TURKISH NEWS ARTICLES

ÖZCAN GÜNDEŞ

BOĞAZİÇİ UNIVERSITY

2020

A DEEP LEARNING-BASED EXTRACTIVE TEXT SUMMARIZATION SYSTEM FOR TURKISH NEWS ARTICLES

Thesis submitted to the

Institute for Graduate Studies in Social Sciences

in partial fulfillment of the requirements for the degree of

Master of Arts

in

Management Information System

by

Özcan Gündeş

Boğaziçi University

2020

DECLARATION OF ORIGINALITY

I, Özcan Gündeş, certify that

- I am the sole author of this thesis and that I have fully acknowledged and documented in my thesis all sources of ideas and words, including digital resources, which have been produced or published by another person or institution;
- this thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- this is a true copy of the thesis approved by my advisor and thesis committee at Boğaziçi University, including final revisions required by them.

Signature.....

Date 03.12.2020

ABSTRACT

A Deep Learning-Based

Extractive Text Summarization System

for Turkish News Articles

The goal of this study is to develop an automated extractive summarization system for Turkish news using pre-trained language models. Pre-trained language models have been applied to wide range Natural Language Processing tasks and achieve state of the art performance results. In this thesis, pre-trained language models for Turkish are applied on extractive summarization task. The proposed model has a pre-trained language model and on top of it, Transformer layers are added to capture document level features and semantic relationships between the sentences in the news articles. Then, these sentences are scored with sigmoid function, which outputs a real value between 0 and 1. To train this model, 2076 news are collected from well-known Turkish news website. After the data collection, each sentence in the articles is labelled as 0 or 1 with a heuristic algorithm. By using these labels, an extractive model is trained. In the test time, Top-5 scoring sentences are combined to generate final summaries. Also, to investigate the effects of hyperparameters, 241 different models, which have different architecture and hyperparameter sets, are run. The best one has achieved 38.38 Rouge-1 F score, 26.8 Rouge-2 F score and 38.04 Rouge-L F score. These scores are promising since they are significantly greater than LEAD-5 baseline, which has 37.49, 26.4 and 37.12 Rouge F scores. For this study, LEAD-5 is very strong baseline since the most significant sentences are placed at the beginning of the news to capture the readers' attention. Therefore, the proposed model shows a good performance for Turkish news dataset.

ÖZET

Türkçe Haber Metinleri için Derin Öğrenme Tabanlı Çıkarıcı Metin Özetleme Sistemi

Bu çalışmanın amacı, Türkçe haberler için önceden eğitilmiş dil modellerini kullanarak otomatik bir çıkarıcı özetleme sistemi geliştirmektir. Önceden eğitilmiş dil modelleri, birçok Doğal Dil İşleme görevinde kullanılmış ve yüksek performans sonuçları başarmıştır. Bu çalışmada, çıkarıcı özetleme görevi için derin öğrenme metotları ile önceden eğitilmiş Türkçe dil modelleri kullanılmıştır. Önerilen mimaride önceden eğitilmiş dil modeli üzerine, haberdeki belge düzeyindeki özellikleri ve cümleler arasındaki anlamsal ilişkileri yakalamak için fazladan Transformer katmanları eklenmiştir. Son olarak, haberde yer alan cümleler 0 ile 1 arasında bir değer üreten sigmoid fonksiyonu ile skorlanmıştır. Bu modeli eğitmek için, bilinen bir Türkçe haber sitesinden 2076 haber metni ilgili özetleriyle birlikte toplanmıştır. Veriler toplandıktan sonra, makalelerdeki her cümle, sezgisel bir algoritma ile 0 veya 1 olarak etiketlenmiş ve bu etiketler kullanılarak, çıkarıcı özetleme sistemi eğitilmiştir. Modeli test ederken ise model tarafından en yüksek skoru alan 5 cümle ile haberin özeti üretilmiştir. Ayrıca hiper parametrelerin etkilerini araştırmak amacıyla farklı hiper parametre setlerine sahip 241 farklı model çalıştırılmıştır. En iyi model 38.38 Rouge-1 F skoru, 26.8 Rouge-2 F skoru ve 38.04 Rouge-L F skoruna ulaşmıştır. Bu skorlar, 37.49, 26.4 ve 37.12 Rouge F skorlarına sahip LEAD-5 bazından önemli ölçüde daha yüksek oldukları için umut vericidir. Bu çalışmada LEAD-5, okuyucuların dikkatini çekmek amacıyla en önemli cümleler haberlerin başına yerleştirildiği için çok güçlü bir baz oluşturuyor. Dolayısıyla, önerilen model, Türkçe haber veri seti için oldukça iyi bir performans göstermektedir.

ACKNOWLEDGEMENTS

First of all, I'd like to thank my thesis advisor, Assist. Prof. Ahmet Onur Durahim, for allocating his time for this research and sharing his knowledge in order to guide me through this research. Then, I would like to thank my jury members, Prof. Aslı Sencer and Prof. Erkay Savaş for accepting to attend my thesis jury, allocating their time to evaluate my thesis and their valuable feedbacks.

I am grateful to my beloved parents and my dearest little brother Berk Gündeş. They always support me for every decision that I have made so far. Without them and their support, I would not achieve anything. They are the most wonderful family.

I would also like to thank my friends, Başak Kalfa and Orkun Kocatürk for being with me throughout this thesis process and sharing their valuable feedbacks. Başak is the one paved my master journey and guided me through all steps. She changed my career. Also, Orkun is always good listener and makes valuable comments for every situation with his wide perspective.

I thank my best friends Ezgi Kurtulmuş, Serhat Say, Çağrı Can and Ekrem Kürtül for making this long journey bearable and enjoyable. It is always relaxing to have fun and share my complaints with them.

My last, biggest, and deepest thanks go to Seden Tezel. She is the most incredible person who makes my life beautiful. During this thesis, she always smiles me with her great comments about epochs, greedy algorithms. I sincerely believe that she will be there for me, for every second in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With the internet being an integral part of daily life, people are exposed to a huge amount of written information. A large amount of textual data is produced at any time through news sites, social media platforms and blog posts. Hence, text summarization can provide a more efficient way to reach significant information that appears in huge amounts of textual data. Executive summaries for business reports, abstracts of academic papers and online newsletters about specific topics are some examples for potential text summarization applications. However, summarizing these textual data manually takes a lot of time.

With the progress of computationally capable hardware and deep learning techniques, automated text summarization systems are receiving much attention by natural language processing (NLP) researchers. These systems aim to generate shorter versions of the original document while preserving its salient and significant information (Cheng and Lapata, 2016). There are two main techniques for summarization tasks: extractive and abstractive. Extractive summarization systems generate summaries by copying and concatenating the most important sentences from the original documents (See et al., 2017), whereas abstractive summarization systems aim to generate novel words and phrases not appeared in the original documents with the help of text rewriting operations such as substitution, reordering (Narayan et al., 2018). Human written summaries are usually produced as abstractive because they are rewritten by preserving main ideas in the original document. Extractive approach is easier and usually produces grammatically and semantically correct sentences by copying sentences directly from

the original document (Nallapati et al., 2016a; See et al., 2017; Dong et al., 2018). In addition, the extractive approach computes faster (Zhong et al., 2020) since it does not perform language generation or rewriting operations. Because of these advantages of extractive approach, most of the previous works have focused on this area.

In the previous works, extractive summarization models generally consist of three main steps. These are representing sentences numerically (sentence representations), scoring sentences one by one based on their importance in the original document (sentence scoring) and finally, selecting top scorer sentences to generate final summary (sentence selection). With the progress of neural networks and deep learning techniques, modern extractive approaches utilize neural network architectures due to their ability to learn continuous feature spaces of inputs in order to learn sentence representations and their relationships with each other. In the training phase, these neural network based models take the sentence level features of the original document as input and as a target, they use binary labels for each sentence to indicate whether they should be included in the final summary or not. In other words, the extractive summarization task is treated as a sequence labelling problem with a binary classification. For example, Cheng and Lapata (2016) obtained sentence representations with Convolutional Neural Networks (CNN) and on top of it, Long Short-Term Memory (LSTM) based Recurrent Neural Network (RNN) was utilized to score and extract sentences. Similarly, Nallapati et al. (2016a) run 2-layer Gated Recurrent Unit (GRU) based RNN to generate final summaries. However, Vaswani et al. (2017) showed that Transformer networks perform better than RNN and CNN in many NLP tasks since Transformers can capture longer term dependencies and run in parallel. By training large Transformer networks with huge dataset, it is possible to learn complex linguistic features and this can boost the

performance in NLP tasks. For example, Devlin et al. (2018) offered Bidirectional Encoder Representations from Transformers (BERT) pre-trained language model. BERT is a masked language model (MLM) and pre-trained with enormous English corpora and leads to state-of-the-art performance results on 11 NLP tasks such as machine translation, question answering and text classification. In the training, authors simply mask 15% of the input tokens and then predict those masked tokens with the aim of learning contextual token representations. In addition to BERT, Clark et al. (2020) proposed the ELECTRA language model. Unlike BERT, ELECTRA is a Replaced Token Detection (RTD) language model. The authors stated that masking only a small portion of the input tokens reduces the amount learned from each sentence and leads to the data inefficiency. Therefore, ELECTRA replaces input tokens with incorrect but reasonable fake ones and then tries to predict and determine which tokens have been replaced with fake ones or remained the same. With this way, they believe that it is possible to learn token representations more effectively compared to BERT and as a result, ELECTRA achieves better performance results.

After these highly capable pre-trained language models' development, the researchers have investigated the effects of these models on extractive summarization. Best performing models for this task are based on these Transformer based pre-trained language models like BERT (Bae et al., 2019; Zhang et al., 2019; Zhong et al., 2019; Liu and Lapata, 2019; Zhong et al., 2020). For example, Liu and Lapata (2019) fine-tuned the BERT model with extra Transformer layers to generate final summaries. To get sentence representations in the original document, they use special token representation, which is [CLS], for each sentence. It is possible to represent sentences

with different representation approaches such as taking the average of BERT outputs of all tokens in each sentence, called as mean pooling.

The NLP community showed great interest in broadening these pre-trained models' limits. For example, Schweter (2020) trained both BERT and ELECTRA architectures, called BERTurk and ELEC**TR**A respectively, with Turkish corpus and published these models as open source. BERTurk model was trained with two different vocabulary sizes, 32K and 128K, which are referred as BERTurk base (32K) and BERTurk base (128K) in this thesis, respectively. The vocabulary size shows the number of different tokens used in the pre-training step.

The study conducted in this thesis aims to perform extractive summarization for Turkish news based on pre-trained language models. The main research question is "how automated extractive summarization can be made for Turkish news?". In this context, the main interests of this study are as follows:

- Investigating the effects of the type of pre-trained language models (MLM with BERTurk and RTD with ELEC**TR**A) on performance results
- Understanding the effect of vocabulary size on performance by utilizing and comparing BERTurk base (32K) and BERTurk base (128K) models
- Observing the effects of different sentence representation approaches by proposing new approach
- Investigating the effect of architectural simplicity/complexity on performance results by putting extra single linear layer and 1, 2 or 3 Transformer layers on top of pre-trained language models

Additionally, the great majority of the summarization datasets in the literature are in English and there is no commonly used Turkish summarization dataset. In this study, Turkish news dataset is proposed for the interested researchers with 2076 news articles with their respective human written summaries.

In this chapter, the main purpose of this research has been introduced. Also, problem definition, possible solution approaches and contributions made by this study have been explained. To mention the chapters that will be covered in the rest of this thesis, Chapter 2 mentions the previous works related to label extraction methods, text representations with pre-trained word embeddings and language models, deep learning methods and extractive summarization models which are the main subjects investigated in this study. Chapter 3 shows and details the methodological steps followed in this study, while Chapter 4 gives the details of the conducted experiments and the performance results obtained in these experiments. Chapter 5 presents the details of conclusion and managerial implications of this thesis topic. Finally, Chapter 7 offers further research opportunities to point out the possible improvement areas in this study.

CHAPTER 2

LITERATURE REVIEW

The goal of this study is to develop a deep learning based automated extractive text summarization system for Turkish news. In this context, the dataset which consists of the main article and related human written (abstractive) summary is collected. To achieve this system, firstly, the sentences in the main article should be labeled based on whether they should be included in the extractive summary, or not by considering their relationships with the human written abstractive summary. This step is fulfilled with label extraction methods. Then, both the main articles and their human written summaries should be converted into numerical representations so that they can be input for the deep learning models. Finally, this input is fed to the deep learning algorithms to train models that output the predicted extractive summaries. For this process, different label extraction methods, text representation methods and deep learning methods for the extractive summarization systems are investigated. In this chapter, previous works in the literature for these topics are mentioned.

2.1 Label extraction methods

Most of the summarization datasets contain abstractive summaries only and hence do not contain the sentence labels which indicate whether the sentences in the original article is included in the extractive summary or not. Therefore, the sentence label extraction from the abstractive summaries is needed to reach ground truth binary labels and train the extractive summarization systems which can be treated as a sequence classification problem. In the mentioned problem, each sentence in the original article is

visited sequentially, and a binary decision is made if the visited sentence should be included in the summary or not, by considering previous decisions.

In the literature, there are some commonly used approaches to this problem. For example, Cheng and Lapata (2016) adopted a rule-based method whether the sentence should be labelled as 1, which means the sentence must be in the summary or as 0, otherwise. They trained a separate supervised classifier with 9000 articles by manually labelling the sentences in each article. The classifier was trained using the following features; the sentence position in the article, the unigram and bigram overlaps between the sentence and the related abstractive summary and lastly, the number of entities appeared in the sentence and the summary. They labelled each sentence individually in the articles to reach ground truth labels with the help of this classifier. Even though this method returns more accurate gold extractive labels, it leads to additional annotation costs (Nallapati et al., 2016a). Moreover, since this method labels the sentences individually, it often generates too many positive labels and this causes the model to overfit the data (Narayan et al., 2018).

Another widely used approach for the label extraction is a greedy algorithm (Nallapati et al., 2016a). In this algorithm, the main idea is that the selected sentences in articles should maximize the ROUGE (Lin, 2004) score with respect to the gold summaries. To reach the binary sentence labels, the authors added one sentence incrementally at a time to the previously selected sentences until the ROUGE score between the generated subset of the selected sentences (oracle summary) and the gold summary does not improve. With this way, the selected sentences are labelled as 1 and the other sentences in articles are labelled as 0. Since the oracle summaries include less sentences, the process does not require additional labelling effort and the method is

computationally cheaper than the Cheng and Lapata (2016) method, it was applied in most of the best performed studies (Liu and Lapata, 2019; Zhong et al., 2020; Guo et al., 2020).

As a widely common evaluation metric of summarization systems, ROUGE (Lin, 2004) is a recall-oriented performance evaluation metric which is widely used in natural language processing tasks, like automatic summarization and machine translation. Since different tasks require different evaluation approaches, ROUGE metric has different settings considering the overlapping of n-grams or subsequence between the text output and the reference text. *ROUGE-N Recall* is the number of overlapping n-gram words over the total number of n-grams in reference summary (Equation 2.1). On the other hand, *ROUGE-N Precision* is the number of overlapping n-gram words over the total number of n-grams in predicted summary (Equation 2.2). Finally, *ROUGE-N F score* is the harmonic mean of recall and precision scores (Equation 2.3).

$$ROUGE - N\ Recall = \frac{\#\ of\ overlapping\ N\ grams\ between\ output\ and\ reference\ summary}{total\ \#\ of\ N\ grams\ in\ reference\ summary} \quad (2.1)$$

$$ROUGE - N\ Precision = \frac{\#\ of\ overlapping\ N\ grams\ between\ output\ and\ reference\ summary}{total\ \#\ of\ N\ grams\ in\ output\ summary} \quad (2.2)$$

$$ROUGE - N\ F\ Score = \frac{2 * (ROUGE - N\ Recall) * (ROUGE - N\ Precision)}{(ROUGE - N\ Recall) + (ROUGE - N\ Precision)} \quad (2.3)$$

To illustrate the ROUGE calculation, imagine that the reference summary (S1) is "*The doctor arrived late because of the traffic*" and the model predicts the output summary (S2) as "*The doctor arrived late due to traffic*". When N parameter is chosen as 2, the 2-grams (bi-grams) in these sentences are obtained as follows:

S1 = [The, doctor], [doctor, arrived], [arrived, late], [late, because], [because, of], [of, the], [the, traffic] – 7 bi-grams

S2 = [The, doctor], [doctor, arrived], [arrived, late], [late, due], [due, to], [to, traffic] – 6 bi-grams

For their intersection, 3 bi-grams are common for both summaries, which are "[The, doctor], [doctor, arrived], [arrived, late]". Finally, ROUGE-2 scores between these two texts are calculated as:

- ROUGE-2 Recall = 3/7 = 0.43
- ROUGE-2 Precision = 3/6 = 0.5
- ROUGE-2 F score = (2*0.43*0.5) / (0.43+0.5) = 0.46

In order to measure fluency between human written and automatically generated summaries, ROUGE-1 and ROUGE-2 scores are used for specific values of n = 1 and n = 2, respectively. Besides, ROUGE-L score is calculated to measure the longest common subsequence overlaps between reference and predicted summary, where n = L. It is applied for assessing informativeness of the generated summaries with respect to human written summaries (Liu and Lapata, 2019).

In this study, ROUGE-1, ROUGE-2, and ROUGE-L scores are applied for the performance evaluation of extractive summarization models. Since it looks for exact n-gram matching between predicted summary and human written summary, it may lead to

low scores even if the compared summaries have the semantically same meaning. But it is still very useful for machine translation and summarization tasks. Lin (2004) proved that the ROUGE scores are highly correlated with the human judgments, especially for single document summarization systems like this study.

2.2 Text representation methods

Natural Language Processing (NLP) aims to give computers reading, understanding, and generating ability for human languages. However, humans use words and sentences for communication, whereas computers and machine learning models are not able to process textual input directly. They can only process numerical inputs. Therefore, the textual inputs must be converted into numerical representations so that the machine learning models can interpret and learn the linguistic structures. In the literature, the mapping of each word or phrase in the textual data to the vector of real numbers was named as embedding. *Embedding* and *representation* terms are used interchangeably in this study. In the aim of creating machine learning models using textual data, texts like characters, words, or sentences should be first converted into numerical representations with the help of these embedding techniques. The selection of the embedding techniques may have an effect on the applied machine learning algorithms' performance for downstream tasks such as text classification, text summarization and machine translation. In the literature, plenty of methods have been proposed so far. In this section, these methods and their advantages and disadvantages are mentioned.

2.2.1 Bag-of-words (BOW) Approach

BOW is one of the most commonly used embedding methods. In this method, each document is represented by the importance of the words in the documents. To determine and measure the importance of these words, the most widely preferred metric is the term frequency-inverse document frequency (TF-IDF) score. The term frequency reveals the number of times which a term occured in a given document; while, inverse document frequency is used to understand how much information a word provides by revealing the number of documents the word appears in. Moreover, the inverse document frequency measures the rareness of the given term or word across all documents. TF-IDF scores are used widely in tasks like search engine ranking, stop-words filtering and text summarization. The calculation of TF-IDF scores is revealed in Equation 2.4.

$$w_{i,j} = tf_{i,j} * \log(\frac{N}{df_i}) \quad (2.4)$$

$w_{i,j} = tf - idf\ weight\ for\ token\ i\ in\ document\ j$

$tf_{i,j} = number\ of\ occurrences\ of\ token\ i\ in\ document\ j$

$df_i = number\ of\ documents\ that\ contain\ token\ i$

$N = total\ number\ of\ documents$

The BoW approach is simple to understand, implement and it achieves great success in many machine learning tasks. However, it leads to a high dimensional feature vector due to the large size of vocabulary. In other words, the size of the document vectors is too large because each document is represented with the number of the times the word occurs in it and the major portion of the words in the vocabulary are not occurred in the related documents. Therefore, each document vector contains lots of zero

values for the words in the vocabulary, which do not appear in the related document. As a result of this, the sparsity problem occurs in BOW methods, especially for large corpora. Also, these methods suffer from ignoring the context, missing the semantic meaning of the words because they do not consider the positions of the words and their neighbors in the documents. To capture semantic meaning of the words and overcome the sparsity problem, semi-supervised techniques for learning word representations by using very large unlabeled data have been studied. With this way, it is possible to obtain the dense, continuous and lower-dimensional vector representations in order to acquire similar vectors for the semantically similar words (Guo et al., 2014).

2.2.2 Pre-trained word embedding models

With the progress in machine learning and computational capability of computers, it becomes possible to utilize neural networks to obtain word embeddings. Mikolov et al. (2013) proposed the Word2Vec technique which was based on one hidden layer simple neural network. With the publication of this study, pre-trained word embedding models gained popularity. In addition, different pre-trained word embedding models like GloVe (Pennington et al., 2014) and fastText (Bojanowski et al., 2017) have been proposed.

As an input, Word2Vec (Mikolov et al., 2013) takes a large text corpus. Due to its semi-supervised nature, it is not required in any labelling process. After taking the input, the model creates a vocabulary which consists of all words in the corpus and represents each word with one hot encoded vector which has the value 1 for the related word index and 0 for the other words. After that step, the center words are paired with their neighbors in the predetermined maximum distance n and so, window size is equal to 2n+1. Therefore, there are 2n neighbors for each center word. This is the automated

labelling process for Word2Vec. Subsequent to this labelling process, the authors proposed two algorithms which are continuous bag-of-words (CBOW) and Skip Gram. By using word-neighbor pairs, the CBOW uses the context words (neighbors) to predict the center word; whereas, Skip Gram predicts the context words (neighbors) by using the center word. The representation of these algorithms with maximum distance equals to 2 and window size 2n+1 equals to 5 can be seen in Figure 1. In the training of this single layer neural network, the size of the hidden layer is the dimension of final word embeddings and it is determined as a hyperparameter.

Each word is represented with two vectors, one of them is obtained when it is a context word and the second one is obtained when it is a center word. After the training process ends, the final hidden layer weights corresponding to each word's context and center word representations is averaged and used as its final vector representation.

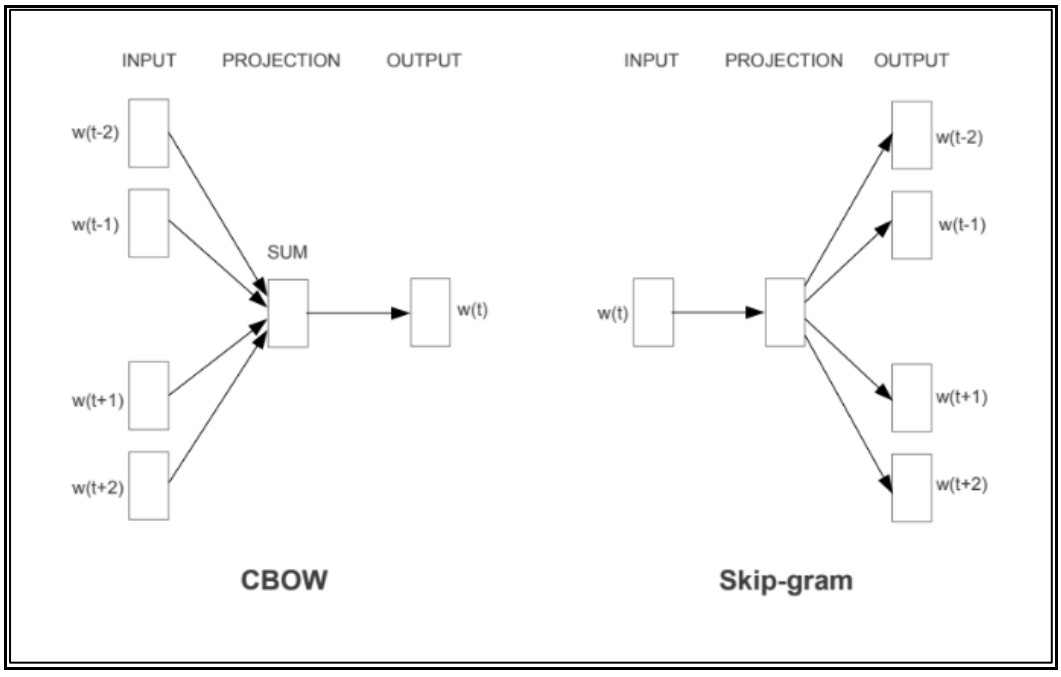

Figure 1. Word2Vec algorithms to learn word representations (Mikolov et al., 2013)

However, Word2Vec does not consider the co-occurrence counts of these word-neighbor pairs in the corpus. At this point, Pennington et al. (2014) proposed the GloVe model by suggesting that the co-occurrence probabilities can encode meaning of the components. This method operates by calculating the co-occurrence count matrix. In this matrix, each row represents the center word, and each column represents the context words (neighbors) that the center word appeared together with. The matrix values state the frequency of the center word with the neighbors in the corpus. GloVe predicts the surrounding word which has the maximum probability among the context words given the center word, by using the log probability of co-occurrence counts to obtain word embeddings.

Both Word2Vec and GloVe have the pre-trained dense and continuous word embeddings, rather than sparse representations as it is seen from Figure 2. In addition, with these methods, the words which are used in the similar contexts have the similar vector representations. Hence, the semantic relationships as well as syntactic similarities can be captured. For example, the distance between the word vectors of "King" and "Man" is quite similar to the distance between word vectors of "Queen" and "Woman" in the vector space.

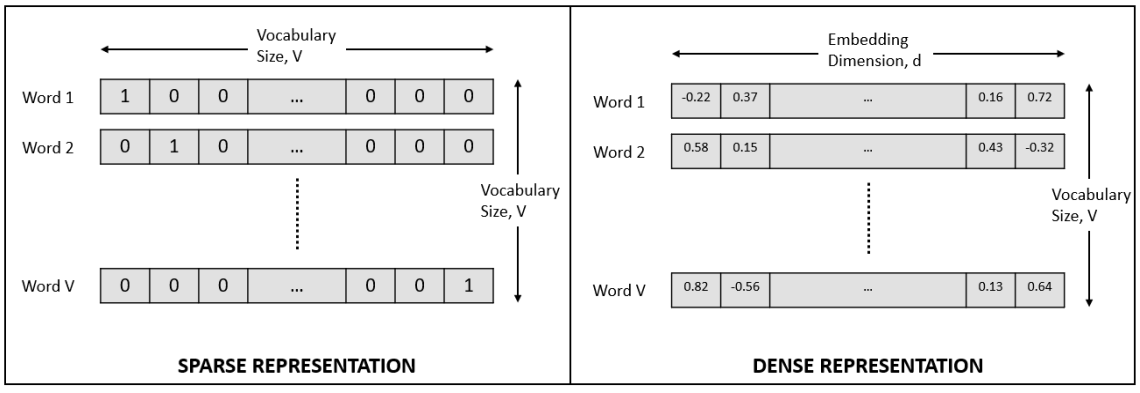

Figure 2. Visualization of sparse and dense representations

These approaches are promising and produce good performance results in most of the NLP tasks such as machine translation, text summarization and text classification. Nevertheless, the rare or misspelled words and the words which are not in the training corpus cannot be represented if that word is not in the training corpus and this leads to out-of-vocabulary (OOV) problems. In other words, when a word which may be new, rare or misspelled from the perspective of the training corpus, is encountered in the inference time, it does not have any proper vector embedding. To overcome this OOV problem, the character-based n-gram level representations were proposed. One of the most popular character based embedding methods was proposed by Bojanowski et al. (2017) and published as an open source library called fastText. FastText has pre-trained word embeddings in more than 100 languages. Similar to Word2Vec, fastText also has a simple neural network and its hidden layer parameters are used to represent words or sentences. But the main difference of fastText from Word2Vec and GloVe, is its character level architecture. With the help of this character level nature, each word or phrase can be represented as n-grams. For example, the word “simple” is divided into 3-gram level as “<si”, “sim”, “imp”, “mpl”, “ple” and “le>” tokens. With this way, the words, which are not included in the training data but encountered in the inference time, can have the proper vector embeddings and the risk of occurring OOV problem is highly minimized.

2.2.3 Pre-trained language models

Even though neural based word embedding methods like Word2Vec, GloVe and fastText have promising performance results in NLP tasks, they do not consider the word orders and positions in the text to obtain the embeddings, also, they suffer from the

non-contextuality. The non-contextuality problem leads to the same vector representation for polysemic words although they have different meanings based on the context. For example, “I have found a solution for the problem.” and “Heat the solution until it becomes clear.” sentences have word "solution" and they have the same vectorial representation, which includes both meanings in the related sentences of the training data, even though they are semantically different.

More advanced (and deeper) neural network architectures that take context of the words into account can deal with this polysemy and non-contextuality problem and hence it can be possible to obtain contextualized word representations. However, training this deeper and larger networks would be costly and require huge text corpora. Many studies have focused on these drawbacks and offered pre-trained language models which are trained with huge corpora and hardware. Pre-trained language models such as ELMO (Peters et al., 2018), BERT (Devlin et al. 2018) and ELECTRA (Clark et al., 2020), can be a strong alternative to get word embeddings by overcoming all previously mentioned problems of sparsity, inability to capture semantic relationship, OOV and non-contextuality.

As a definition, language modelling is the task of assigning probability to sequences by assigning a probability to each token (characters, subwords or words) in a related sequence with respect to the previous tokens (Goldberg, 2017). In other words, language models are trained with the aim of predicting the next word given a sequence of previous words. The common applications of language models are text generation, machine translation and spelling correction. Since language models consider previous tokens to determine the next token’s probability, they are unidirectional, which is left to right, inherently. However, it is important to learn from both directions to obtain

contextual word embeddings. For this purpose, Peters et al. (2018) published ELMO (Embeddings from Language Models). ELMO utilizes 2 bidirectional (both left to right and right to left) LSTM (long short-term memory) architectures to reach contextual word representations. LSTM (Hochreiter and Schmidhuber, 1997) is the special kind of recurrent neural network (RNN). The main advantages of LSTM over standard RNN are its ability to learn long-term dependencies, which is important to remember the previously seen important words and to forget the insignificant ones to process the last ones for the long texts, and to overcome vanishing gradient problem, which occurs in the backpropagation step to update model parameters by calculating gradients in standard RNNs for the long texts. The deep biLSTM layers allow ELMO to learn the contextual meaning of the words in the higher layers and syntactic relationships in the lower layers. In addition to LSTM layers, ELMO utilizes character level convolutions rather than word level training to overcome OOV problems. As a result, ELMO is trained for 10 epochs on 1B Word Benchmark dataset (Chelba et al., 2014) with 93.6 million parameters where the hidden size of biLSTM modules are 4096 and the dimension of the final embeddings is 512.

Although the specialized RNN network architectures like LSTMs and GRUs (Gated Recurrent Units) (Cho et al., 2014) proves their effectiveness over the performance metrics of many NLP tasks, they suffer from the inability to parallelize. This inability leads to huge memory limitations, computational complexity, and training time with a large text corpus. In 2017, Vaswani et al. (2017) proposed a new network architecture called Transformers and they achieved great results in machine translation and constituency parsing with smaller training costs. Transformers are more parallelizable than LSTMs and require significantly less training time and also, they can

handle long range dependencies easily like LSTMs. The main idea behind them is to handle input and output dependencies with attention and recurrence. The Transformer's architecture (Vaswani et al., 2017) can be seen in Figure 3. The architecture seen on the left half of Figure 3 is the encoder of the Transformer and the right half is the decoder part. The number of encoder and decoder units in one Transformer block is the hyperparameter and in Vaswani et al. (2017), the number of encoder and decoder units have been chosen as 6. Each encoder is identical and stacked on top of the previous one. Similarly, each decoder is identical to other decoders and stacked on top of previous ones. As a working principle, word embeddings and positions of the words in the input sequence are passed to the first encoder. With the help of multi-head attention and feed forward structures, they are transformed and moved forward to the next encoder. Then, the last encoder's output is passed to all decoders' multi-head attention parts. Multi-head attention refers to the computation of multiple self-attention in parallel. Self-attention, also known as intra attention, is the mechanism of relating different positions in a single sequence to be able to compute a representation of the whole sequence. These relations are calculated with scaled dot products between the words in the sequences in (Vaswani et al., 2017).

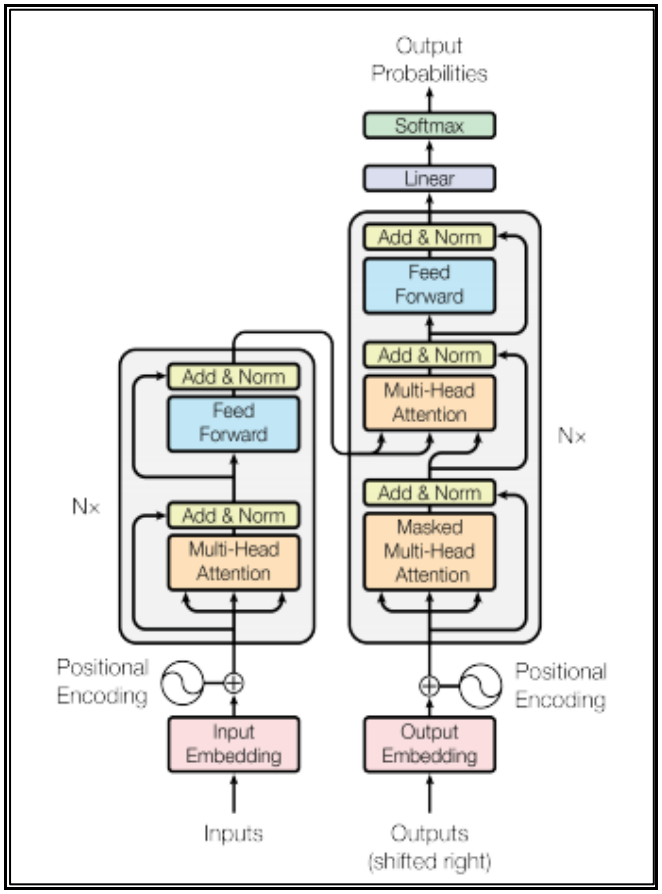

Figure 3. Transformer model's architecture (Vaswani et al., 2017)

With the invention of Transformers, NLP studies accelerated and lots of state of the art (SoTA) performance results were achieved with the models that incorporate Transformers in most NLP tasks. One of the most promising models for the language modeling task is the BERT (Devlin et al., 2018). BERT, which stands for Bidirectional Encoder Representations from Transformers, is also bidirectional like ELMO as the name implies. But unlike ELMO, it utilizes Transformer networks instead of LSTMs and is trained with the subword level tokens rather than character level. For input representation, BERT uses token, segment and position embeddings of the related token and represent this token by summing up these 3 types of embeddings as seen in Figure 4.

This representation can present both a single sentence (in this case, each token's segment embeddings are the same) and a pair of sentences (in this case, for the tokens which belong to different sentences, they have different types of segment embeddings as seen in Figure 4) such as question-answers, and machine translation from one language to another. In this architecture, the first token of every sequence is [CLS] which is the special classification token. [CLS] token can be used as the aggregate sequence representation for classification tasks since every token in the sequence is related with different positions due to self-attention structure in the Transformer blocks. Also, [SEP] token is a special token used as a separator token (e.g. separating questions with their answers).

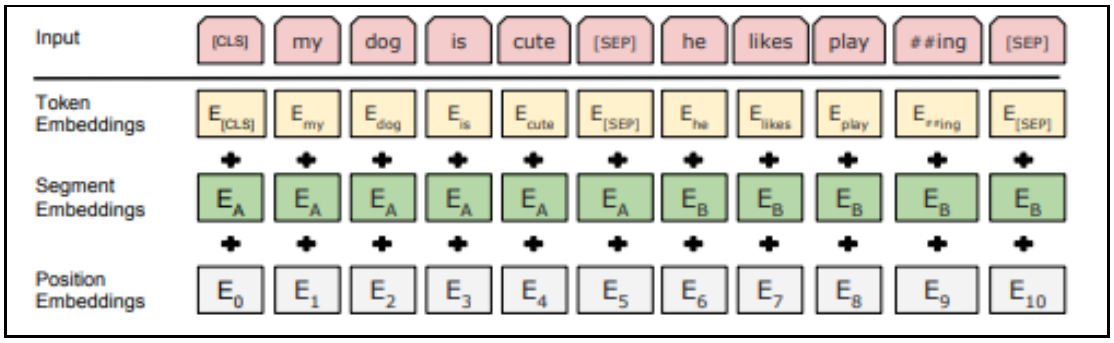

Figure 4. BERT input representations (Devlin et al., 2018)

In the pre-training phase, the input representations of the tokens are passed into the bidirectional Transformer encoder blocks. During the training, authors simply mask 15% of the input and then predict those masked tokens similar to the language modelling. Therefore, this way of approaching to solve the language modelling task is called masked language modelling (MLM). For the pre-training corpus, the English model was trained with the BooksCorpus (Zhu et al., 2015), which has 800 million

words, and English Wikipedia, which has 2,500 million words. Authors published 2 trained BERT models, Bert-base and Bert-large. In the base model of BERT, there are 12 layers with 768 hidden sizes, 12 self-attention heads and totally 110 million parameters, whereas in the large model, there are 24 layers with 1024 hidden sizes, 16 self-attention heads and totally 340 million parameters. Vocabulary size which is the number of subwords in the training corpus is around 30K for both Bert-base and Bert-large models. To fine-tune these BERT models for the downstream NLP tasks such as text classification, question answering and part-of-speech (POS) tagging, the only requirement is the labelled dataset for these tasks. Fine-tuning BERT with the task related datasets leads to very high performance results according to the study. Therefore, fine-tuning the BERT and obtaining the sentence and token representations by using the BERT model as the textual feature extractor is quite common in the previous NLP studies. The pre-training and fine-tuning for classification task procedures of the BERT are shown in Figure 5. For example, for sentiment analysis task, the input text and related sentiment were fed into the BERT model and [CLS] token representations were used to reach the sentence embedding. By using a single linear classification layer over these [CLS] representations, the model can be fine-tuned where the corresponding model parameters are updated. When fine-tuning the BERT for a specific task, parameters in BERT are jointly fine-tuned/updated with additional task specific parameters, unlike ELMo, whose parameters are usually fixed and only the task specific parameters are learned (Liu and Lapata, 2019).

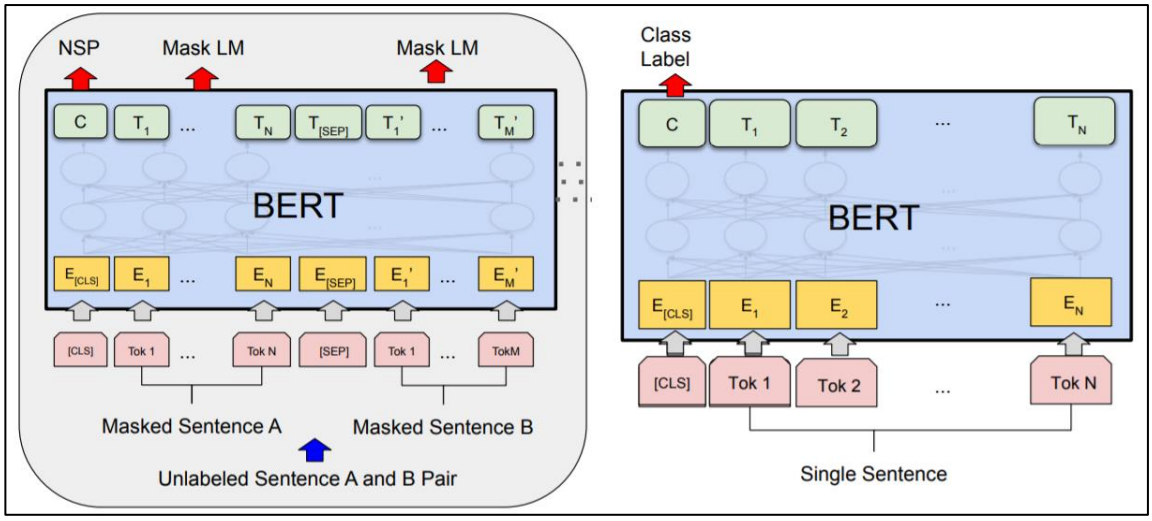

Figure 5. Pre-training (left) and fine-tuning (right) procedures of BERT

Even though the BERT model has become one of the major breakthroughs in NLP research, there are some known limitations. According to the Devlin et al. (2018), their BERT model is creating a discord between pre-training and fine-tuning because [MASK] tokens are never seen during the fine-tuning phase. These tokens are utilized only in the training time to train the language model. In addition to this mismatch, masked language modelling in BERT has been masking only 15% of tokens in each sentence and this reduced the amount learned from each sentence leading to the data inefficiency. To address these problems, Clark et al. (2020) presented the ELECTRA, which stands for Efficiently Learning and Encoder that Classifies Token Replacements Accurately, and as its name implies, it uses a different approach to pre-train language models with the aim of providing the benefits of BERT in a more data efficient manner. Unlike BERT, ELECTRA uses replaced token detection (RTD) instead of MLM. The RTD trains a bidirectional model like MLM but learns from all input positions. To accomplish this, instead of replacing tokens with masking them as in BERT,

ELECTRA's generator model replaces input tokens with incorrect but reasonable fake ones via its small language model. Then, the discriminator model tries to predict and determine which tokens in the original sequence have been replaced with fake ones or remained the same. The simplified version of ELECTRA's working principle is shown at Figure 6. Since binary classification is performed over every input token rather than only 15% of all input tokens as in BERT, ELECTRA can achieve the same performance by using fewer examples. The main reason for this efficiency increase is that the ELECTRA takes more signal per example (Clark et al., 2020). After the training phase, the generator part is dropped, and the discriminator model is ready to use for fine-tuning. The base model of ELECTRA has the same number of layers (12), hidden size (768) and parameters (110 million) with the BERT base model. But ELECTRA base models' performance scores are higher than the BERT large model, as shown in the ELECTRA study, (Clark et al., 2020). This proves their efficiency claims.

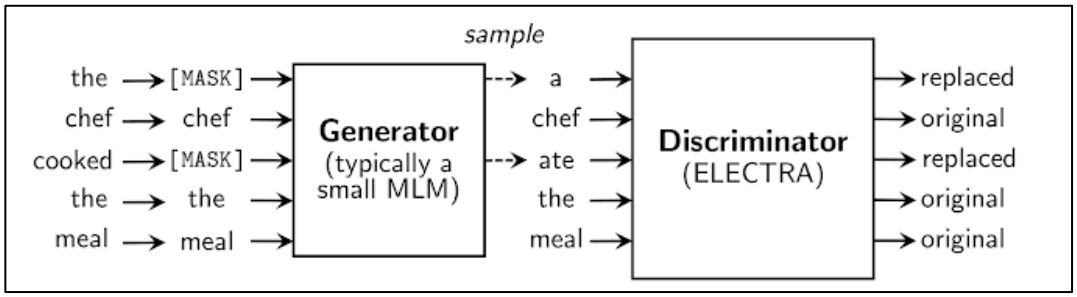

Figure 6. The overview of replaced token detection approach (Clark et al., 2020)

Both BERT (Devlin et al., 2018) and ELECTRA (Clark et al., 2020) language models were trained with English text corpus. Since their source codes are open, the NLP community trained these models for different languages. For example, Schweter (2020) presented the BERT and ELECTRA models for Turkish, called BERTurk and

ELEC**TR**A, respectively. These models have the same architecture with original ones and the only difference from them is the language and size of the training corpora. Both models were trained with 35GB Turkish textual data, which has 4,4 million tokens. The data includes Turkish OSCAR corpus, Wikipedia, various OPUS corpora and a special corpus provided by the community. As a result of the BERTurk training process, 2 different pre-trained BERTurk models were published. The difference between the models is the vocabulary size. The vocabulary size, that is the number of word-piece tokens, of the BERTurk base (32K) model is 32K, whereas the vocabulary size of the BERTurk base (128K) model is 128K.

2.3 Deep learning models for extractive summarization

Automated text summarization gains popularity with the progress of computer capabilities and the emergence of deep artificial neural networks. With the help of the deep learning models which consist of neural networks with several layers, the higher-level textual features such as interrelation between sentences and the semantic understanding of the sentences can be learned.

Extractive summarization systems generate the summary by determining and selecting the most significant and salient sentences in the input document by scoring them with different techniques. Several studies about the extractive summarization were proposed in the literature. However, a major portion of these applied over the English texts. There is a considerably small number of studies focused on the low resource languages such as Turkish. The most commonly used dataset for the multi-sentence summary text summarization is the CNN/Daily Mail dataset. This dataset consists of 311,971 English news with their related multi-sentence human written (abstractive)

summaries (Nallapati et al., 2016b). The data has been preprocessed so that each entity occurrences are replaced with document-specific integer-ids beginning from 0 to decrease the vocabulary size. As the dataset statistics, it has 286,817 (92% of all dataset) training articles, 13,368 (4% of all dataset) validation articles and 11,487 (4% of all dataset) test articles. The main articles in the training set have 766 words with 29.74 sentences on average, whereas the human written summaries consist of 53 words and 3.72 sentences on average. Moreover, there is a leaderboard for this dataset to show the performance results of the competitive extractive summarization approaches (Ruder, 2020). The best performing methods are sorted by their final ROUGE-L F scores obtained from the test set.

One of the earliest extractive summarization approaches over this dataset belongs to Nallapati et al. (2016a). They treated the extractive summarization as a sequence classification problem. For this problem, each sentence is scored with respect to the probability of being included in the final summary based on the sentence features, which are its content richness, its salience with respect to the main article, its novelty with respect to the previously selected sentences to form the extractive summary and lastly, its position in the main article. To train the model with this approach, each sentence in the main article needs to be labelled as 0 or 1 depending on whether that sentence should be excluded or included in the final summary, respectively. For this labelling process, they proposed a greedy algorithm as mentioned in Section 2.1.

Their model, called SummaRuNNer, has 2-layer bidirectional GRU networks. Its input is the 100-dimensional Word2Vec embeddings of each word in each sentence. The bottom layer takes these input representations and runs at the word level to compute hidden state representations for each word by considering representations of previous

and next words. This layer outputs each word's hidden state representations and by concatenating them and taking average over the word count for each sentence in the main article, the initial sentence representations are obtained. The upper biGRU layer runs over the sentences by taking the initial sentence representations outputted from the bottom layer. The hidden states of this upper layer encode the contextual sentence representations. Finally, one logistic layer makes a binary decision over these representations to calculate the cross-entropy loss (Equation 2.5) by considering whether the selected sentences by the SummaRuNNer are correctly chosen based on their labels. In the inference time, each sentence in the main article belonging to the test set articles is scored by the model and top 3 scored sentences are chosen to generate the final summaries. Finally, they achieved a 35.5 ROUGE-L F score.

$$CE = -\frac{1}{N} * \sum_{i}^{N} y_i * \log(t_i) + (1 - y_i) * \log(1 - t_i) \quad (2.5)$$

$CE = cross\ entropy\ loss$

$y_i = actual\ (true)\ label\ for\ sample\ i$

$t_i = predicted\ label\ for\ sample\ i$

$N = total\ number\ of\ samples\ in\ the\ dataset$

In 2017, See et al. (2017) proposed the non-anonymized version of CNN/Daily Mail. The only difference of this dataset from the previous anonymized one is that the entities in the main articles are not replaced with any values. The main reason that Nallapati et al. (2016b) replaced the entities with unique integers was to decrease vocabulary size. But See et al. (2017) stated that using non-anonymized (original) dataset is more favorable because it does not require a preprocessing step. Even though

See et al. (2017) focused on abstractive summarization and there is no extractive model proposed in their study, the authors showed that simply selecting the first 3 sentences from the main article to generate the final summaries (LEAD-3) leads to a 36.6 ROUGE-L F score. LEAD-3 is often used as a baseline in the extractive summarization field. In the following works, the non-anonymized version of CNN/Daily Mail dataset is used.

Training extractive summarization models with the cross entropy loss (Equation 2.5) and evaluating the performance of models with ROUGE scores renders a mismatch between what is being optimized and what is being used as the performance metric (Narayan et al., 2018) and also, the cross entropy loss may cause underfitting since it only maximizes the probabilities for the sentences labelled as 1 and ignores all 0 labelled sentences (Bae et al., 2019). To globally optimize ROUGE metrics, Narayan et al. (2018) proposed REFRESH model which applies reinforcement learning to achieve optimal ROUGE scores. The words in each sentence in the main articles are initialized with 200 dimensional Word2Vec embeddings and passed into a convolutional neural network (CNN) to obtain the sentence representations. After that, these sentence representations are fed into the LSTM network, and then the output of the LSTM network is used as the main article representation. On top of this LSTM layer, one more LSTM network is put to read a sentence representation from the CNN layer and make a binary prediction for this sentence conditioned on the main article representation taken from the former LSTM layer and the selected sentences in the previous time steps. Then, the result of binary predictions is used to rank sentences with the softmax layer's scores. Higher rank means higher possibility of being in the final summary. To train this model, reinforcement learning based objective function is used with the aim of generating a

final summary that should have the maximum ROUGE score based on the actual human written summary. The training process took around 12 hours on a single GPU. Finally, REFRESH achieves a 36.6 ROUGE-L F score which is equal to the baseline LEAD-3 approach.

With the emergence of Transformer architecture (Vaswani et al., 2017) and pre-trained language models, the extractive summarization approaches got more capable of producing better performances like the other NLP tasks. In 2019, Zhang et al. (2019) proposed HIBERT, which stands for hierarchical bidirectional encoder representations from Transformers. The authors stated that the pre-trained models like BERT (Devlin et al., 2018) aim to pre-train in word level contextual embeddings based on the sentence words appeared in. However, HIBERT aims to pre-train hierarchical document encoders for the summarization task since it requires document level encodings rather than sentence level. HIBERT is inspired by the original BERT model but it has three Transformer networks, two of them used for encoding and last one is allocated for decoding, rather than single layer Transformer used in BERT. The first Transformer network runs over word level to encode sentences and the second one runs over the sentence representations outputted from the first one in order to come up with the document representations. In this second Transformer network, some sentences are masked, similar to the word masking used in BERT, and these masked sentences are predicted by the third (decoder) Transformer network. Instead of fine-tuning the BERT model, the authors trained HIBERT in an unsupervised manner from scratch with GIGA-CM dataset which has almost 7 million documents. This training process took around 20 hours for each epoch with 8 NVIDIA Tesla V100 16GB GPUs and in total the model is run for 45 epochs. After the training process was completed, they fine-tuned

the model for the extractive summarization with CNN/Daily Mail. After obtaining the main document based contextual sentence representations, they put one simple linear layer with softmax function to reach the probabilities of the sentences being in the final summary. These probability scores are compared with the true labels obtained from the greedy algorithm (Nallapati et al., 2016a) and the cross-entropy loss is calculated to update the model parameters. With this pre-training approach, training a new language model from scratch like BERT and then fine-tuning it with the CNN/Daily Mail dataset, they achieved 38.83 ROUGE-L F score which is a significant improvement as compared to the scores of the previous works.

Training of these large language models from scratch can be very costly and time consuming. As stated, HIBERT was trained around 900 hours with eight 16GB GPUs. Therefore, fine-tuning pre-trained language models by manipulating them based on the task is a more feasible solution with respect to computational cost. For example, Liu and Lapata (2019) proposed the BERTSumExt method within which they encode and manipulate multi-sentential inputs and proposed a novel BERT architecture for extractive summarization called BERTSUM. They have added external [CLS] tokens to the beginning of each individual sentence in the main article together with using interval segment embeddings in order to differentiate those multiple sentences in the document. Interval segment embeddings were assigned considering the order of the sentences. For example, segment embedding for the sentence(i) is assigned depending on if index i is odd or even. These changes made over the original BERT architecture can be seen in Figure 7. The architecture on the left of the figure shows the input formation of the original BERT method, whereas the architecture on right illustrates the BERTSUM

settings. In BERTSUM, the green items show the interval segment embeddings created to distinguish the sentences.

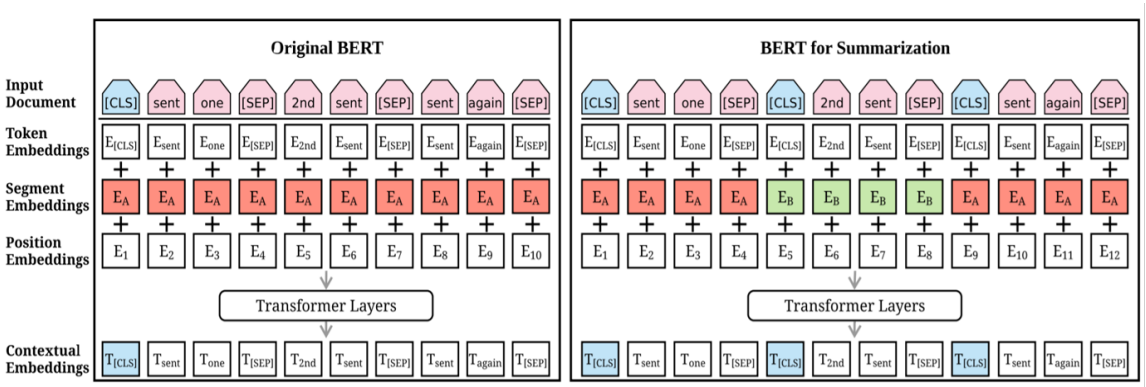

Figure 7. Architecture of original BERT and BERTSUM model (Liu and Lapata, 2019)

The BERTSUM model takes the embeddings of the words in the main articles computed considering the segment and position embeddings as input and by passing them into the BERT Transformer layers, it outputs the token representations as seen in Figure 7. By using each CLS token, it is possible to reach each sentence representation in the main article. Due to the self-attention mechanisms in the Transformer networks, the CLS tokens can encode information obtained from all the tokens in the same sentence.

The authors added several inter-sentence Transformer layers on top of the BERTSUM to capture document level features. These layers take the CLS token embeddings as the related sentence representations, calculate document level features and output the learned document specific sentence representations. Finally, these representations are classified with the sigmoid classifier to determine their labels. Based on the experiments to determine the optimum number of extra Transformer layers, 2 extra layers performed best. They called this final model as BertSumExt. In the training,

cross entropy loss function was applied. During the prediction of sentences of the summary, they have used this model to calculate scores of each sentence and then rank them from highest to lowest, and finally, select the top scored 3 sentences. In addition to the selection phase, they have applied a method called Trigram Blocking which ensures that there is no trigram overlapping between the summary and the candidate sentence to reduce redundancy as a post processing step. This model has reached 39.63 ROUGE-L F score by using the base version of BERT and 39.90 score with the large version of BERT.

Even though the BertSumExt produced high performance results and took the leadership in the extractive summarization leaderboard for CNN/Daily Mail dataset, it works at the sentence level. That is, although the sentence representations can reflect document level features, they are chosen individually based on their scores at the end. Zhong et al. (2020) addressed this problem with their MatchSum approach. They stated that the output summaries consist of the individual sentences having the highest ROUGE scores in the sentence level studies. However, the summary containing these sentences may not be the optimal candidate summary for the related main article. Therefore, it is needed to use summary level extractors rather than sentence level ones for the extractive summarization. Unlike sentence level extractors, summary level ones choose the best candidate summary based on their ROUGE scores with human written summary and it may not contain the highest scorer sentences.

The authors formulated their approach as semantic text matching. In this approach, better candidate summaries should be semantically closer to the main article, while the human written summary should be the closest one. This approach can generate better summaries as compared to sentence level approaches. To support this, they

conducted experiments over different datasets and they found out that if the summaries are too short like 25-30 words or if the summaries are too long like 200+ words, their approach does not lead to much improvement. However, for a medium length summary around 50-100 words like the ones in the CNN/Daily Mail dataset, the summary level approach can be rewarding.

They have formulated the problem by first constituting all possible N sentence combinations of the sentences in main articles to generate all possible candidate summaries. For example, N is set to 2 and 3 for the CNN/Daily Mail dataset. After the candidate summaries are determined, their ROUGE scores, which are based on the human written (abstractive) summary, are calculated and they are sorted in descending order. Then, the candidate summary and the main article are fed into the Siamese BERT network, which consists of two BERTs with tied-weights and a cosine similarity layer in the inference phase. To determine the cosine similarity, the CLS token representations of both main article and candidate summary is used. As mentioned previously, the best candidate summary should have the highest similarity score compared to the other ones, while the human written summary should have the higher similarity score with respect to the main article than all candidate summaries. To guarantee these constraints (Equation 2.7 and 2.6, respectively), the authors used a margin-based triplet loss function (Equation 2.8) to update the weights of the Siamese BERT network. In Equation 2.6, the first function f calculates the cosine similarity between the main article D and candidate summary C, whereas the second one calculates the cosine similarity between the main article D and human written summary C*. With margin $\gamma 1$, it is aimed that the second similarity score should be higher than the first score. In Equation 2.7, firstly all candidate summaries are sorted in descending order of ROUGE scores with the human

written summary. The candidate pair with a larger ranking gap should have a larger margin, which is (j-i)*γ2. Also, the higher scorer candidate summary, Ci should have a higher similarity score based on the main article D compared to lower scorer candidate summary Cj. Finally, in Equation 2.8, the losses, which are calculated in Equation 2.6 and Equation 2.7, are summed up to obtain the total loss.

$$\mathcal{L}_1 = \max(0, f(D, C) - f(D, C^*) + \gamma_1) \quad (2.6)$$

$$\mathcal{L}_2 = \max(0, f(D, C_j) - F(D, C_i) + (j - i) * \gamma_2 \;\; (i < j) \quad (2.7)$$

$$Total\ Loss = \mathcal{L}_1 + \mathcal{L}_2 \quad (2.8)$$

Although this matching idea is quite intuitive, it suffers from considering the large number of candidate summaries. For example, CNN/Daily Mail articles have 30 sentences on average and taking both 2 and 3 sentence combinations of them results in 4495 possible candidate summaries for each article. To handle this problem, the authors prune the documents by selecting the most K salient sentences in the main articles and they have chosen K as equal to 5 for the CNN/Daily Mail dataset. So, for each main article, they chose 5 sentences and took 2 and 3 sentences of them to generate 20 candidate summaries in total for each main article. To prune the main articles, they employed BertSumExt (Liu and Lapata, 2019) approach to calculate sentence scores and then, they obtained the top 5 scoring sentences and used them to generate the candidate summaries.

They trained MatchSum model's Siamese Bert networks with eight Tesla V100-16GB GPUs and the training took around 30 hours. In the inference time, the main

articles' candidate summaries are generated and based on the Siamese Bert networks, their vector representations are calculated. Finally, the one with the highest cosine similarity with the main article's vector representation is chosen as final extractive summary. The MatchSum with Siamese Bert achieved a 40.38 ROUGE-L F score. After obtaining this score, the authors change Bert encoder in the Siamese network with RoBERTa (Liu et al., 2019) encoder which is also a pre-trained language model similar to BERT, but it is pre-trained with 63 million English news. As a result of this change, MatchSum achieved slightly higher performance score in CNN/Daily Mail which is 40.55 ROUGE-L F score. The authors explained this improvement as the similarity between the training corpus of RoBERTa and the fine-tuning dataset which is CNN/Daily Mail news. As a result, 40.55 ROUGE-L F score is the highest score achieved for CNN/Daily Mail dataset.

With the aim of achieving promising performance scores for the Turkish news dataset gathered for this thesis, the different Transformer architectures proposed in the literature are manipulated with different settings and hyperparameters.

CHAPTER 3

RESEARCH METHODOLOGY

In this section, detailed explanations of the approach followed in the extractive summarization for Turkish news in this thesis are given. Firstly, the data collection steps are given, and the collected data is investigated with descriptive statistics after performing data preprocessing steps. Secondly, the label extraction method that is applied to obtain gold label extractive summaries from the human written (abstractive) summaries is described. Then, input data preparation steps taken in order to make raw data suitable for the proposed models are stated. Finally, the details of the proposed summarization model are explained. The overview of the research methodology followed in this thesis can be seen in Figure 8.

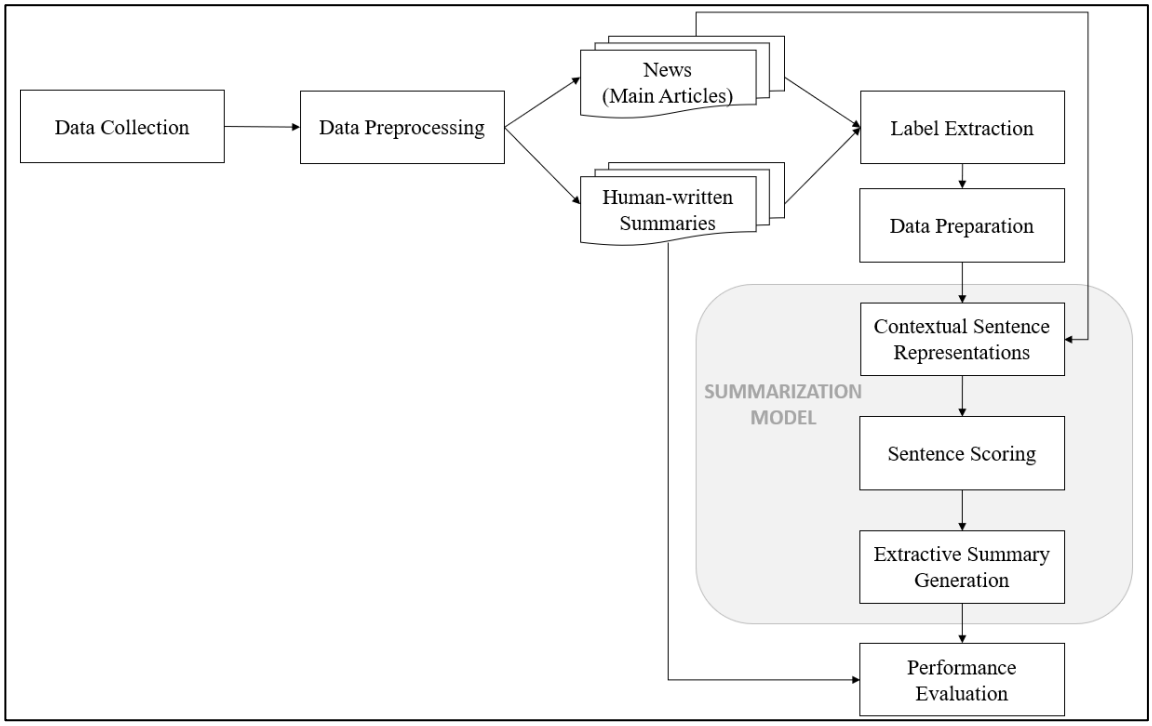

Figure 8. Overview of the research methodology

3.1 Data collection and preprocessing

Since most of the summarization corpora is in English and there is no suitable multi-sentence summarization dataset available in Turkish in the literature, a new corpus for Turkish news summarization is required to be created. In order to do so, the well-known news website, which has both long texts together with their human written (abstractive) summaries was discovered as seen in Table 1.

In this thesis, a news website was crawled to constitute Turkish news dataset by using the Python Selenium library. Then, the collected texts (both main articles and their respective human written abstractive summaries) were preprocessed. The applied preprocessing steps were converting text to lowercase, removing URLs, hashtags, some special characters such as "|", stripping off the excess white spaces in order to help the tokenization process to tokenize the sentences in the texts properly and finally, dropping duplicate news if they have the same content. As a result of the preprocessing steps, the final dataset has 2076 news and the related statistics regarding the dataset can be seen in Table 2. The average news length is almost 20 sentences with 359 words and the average human written abstractive summary length is almost 5 sentences with 84 words. Both word and sentence level compression ratios, defined as the length of news divided by the length of summaries, are quite similar to each other which are 4.24 and 4.26, respectively. Additionally, the dataset novelty, which is the percentage of bi-grams in the gold abstractive summary that are not included in the related article, is 34.83%. This statistic is a proxy for the abstractiveness and shows the suitability of the dataset in terms of extractive or abstractive summarization (Scialom et al., 2020). Based on this, the dataset is highly suitable for extractive settings since the novelty is quite low. For comparison, novelty of the most widely used dataset CNN/Daily Mail is around 52%

(Liu & Lapata, 2019). Finally, the Turkish news dataset has almost 72K different words and 8K of them are occurring more than 10 times.

Table 1. Sample News with the Corresponding Summary (Sonmez, 2016)

Main Article	Facebook'un CEO'su Mark Zuckerberg İspanya / Barselona'daki Mobile World Congress kapsamında düzenlenen Samsung etkinliğinde sahne aldı. Sanal gerçeklik ile ilgili bir sunum yapan Zuckerberg Facebook'un geleceğinin yapay zekada yattığını söyledi. Sanal gerçeklikle ilgili "Sizi sarmalayacak, insanları bir araya getirecek ve bütün bunlar düşündüğünüzden çok daha yakında gerçekleşecek" şeklinde konuşan Zuckerberg "Uzun süredir bu deneyimi insanlara yaşatabilmeyi bekliyordum, şimdi o gün geldi" şeklinde heyecanını dile getirdi. Pazar günü düzenlenen Samsung etkinliğinde, Facebook ve Samsung'un ortak çalışmasının ürünü olan 360 derece fotoğraf ve video kaydı yapabilen Gear 360 tüketiciye tanıtıldı. Geçtiğimiz Kasım ayında Samsung 360 derece video izlemeye olanak sağlayan bir sanal gerçeklik cihazını tanıtmıştı. Gear VR adlı bu ürün Facebook'un 2014'te 2 milyar dolar gibi iddialı bir bedel ödeyerek bünyesine kattığı sanal gerçeklik şirketi Oculus'un teknolojisini kullanıyor. 99 dolara satışa sunulan ürün, içine (ekran ve ana işlemci görevi üstlenen) uyumlu bir Samsung telefon yerleştirilerek kullanılıyor. Zuckerberg "Sanal gerçeklik herkesin, istediği her şeyi üretip deneyimleyebileceği yeni platform. Çok yakında herkes, sahnelerin tamamını sanki oradaymış gibi deneyimleyebilecek" şeklinde hayalini özetledi.
Human-written Summary	Barcelona'da gerçekleştirilen basın toplantısında Facebook-Samsung ortak çalışmasının ürünü 'Gear 360' tanıtıldı. Oculus ve Facebook'un sahibi Mark Zuckerberg'e göre sanal gerçeklik sayesinde çok yakında yepyeni bir dönem başlayacak.

Table 2. Turkish News Dataset Statistics

Dataset Size	2,076
Training/Valid/Test Sets Size	1,476/300/300
Sentence-level News Length	19.96 ± 9.81
Sentence-level Summary Length	4.71 ± 2.45
Sentence-level Compression Ratio	4.24
Word-level News Length	359.06 ± 168.49
Word-level Summary Length	84.29 ± 34.36
Word-level Compression Ratio	4.26
Novel bi-grams in Gold Summary	34.83%
Total Vocabulary Size	71,547
Total Vocabulary Size (Occurring 10+ Times)	8,243

3.2 Label extraction

Like the other widely used summarization datasets, the collected dataset does not contain the sentence labels which indicates whether the sentence is included in the extractive summary. Therefore, the sentence label extraction from the abstractive summaries is needed to reach ground truth binary labels to train the extractive summarization systems as mentioned in Section 2.1.

In order to avoid extra annotation costs and the overfitting risk, the rule-based label extraction method proposed by Cheng and Lapata (2016) is not preferred as the

label extraction method in this thesis. Rather, the greedy algorithm (Nallapati et al., 2016a) is applied since it is cheaper and preferred by the best systems proposed in the literature.

To implement this algorithm, the original articles are tokenized into sentences by using the Python nltk package (Bird, Klein and Loper, 2009). Then, the sentence in the article which has the highest ROUGE-2 F1 score with respect to the related gold abstractive summary is selected. Following that, the remaining sentences are added to the first chosen sentence, one at a time until the ROUGE-2 F1 score does not improve anymore. Finally, sentences selected in this way by the greedy algorithm are labelled as 1 and the remaining ones are labelled as 0. Additionally, selected sentences are combined to constitute the ORACLE (gold extractive) summaries which will be used to compute the upper bound for the performances of the extractive summarization models.

After the label extraction step, the dataset contains both the original article, abstractive summary, and extractive sentence labels. To see the most salient sentences' positions in the original article based on the extracted labels, the sentence positions with respect to the percentage of those sentences included in the respective oracle summaries are plotted and shown in Figure 9. As seen, there is no uniform distribution between the positions. Almost 68% of the first sentences and 59% of the second sentences in the input document are included in the oracle summaries. This observation supports the claims of See et al. (2017) about the news datasets that the news articles are generally structured with the most significant and salient sentences at the beginning in order to get attention from the readers. Therefore, Lead-N, which is selecting the top-N sentences from the news, is the most basic but effective approach for the extractive summarization systems developed for the news. Since in the major portion of the oracle summaries in

the dataset, the number of sentences is in range between 0-5, as seen in Figure 10, and the average sentence length of the gold summaries is almost 5, the performance Lead-5 approach is taken as the lower bound for the extractive summarization models.

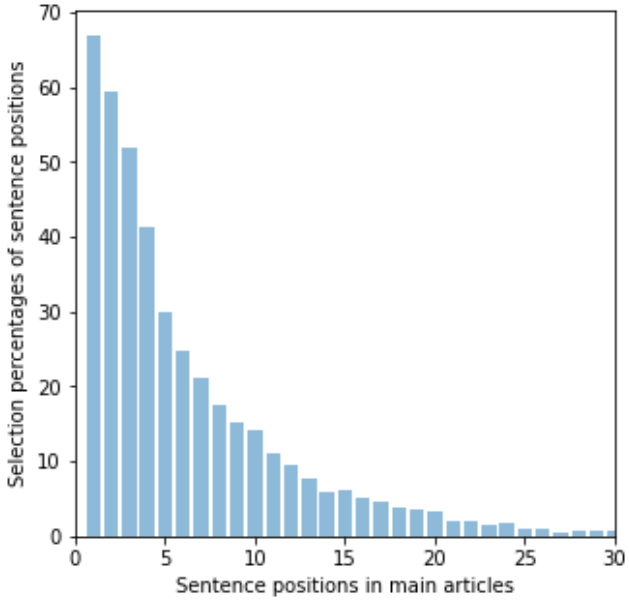

Figure 9. Selection percentages of sentence positions in the main articles

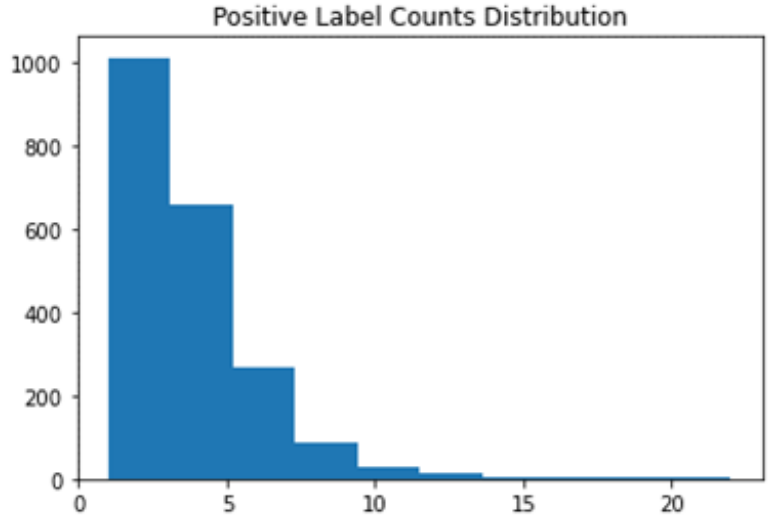

Figure 10. The sentence length histogram of the oracle summaries

3.3 Input data preparation for Transformer based pre-trained sentence encoders

Transformer based pretrained language models like BERT (Devlin et al., 2018) and ELECTRA (Clark et al., 2020) have been applied in most of the NLP tasks as encoders to represent words and sentences. However, summarization requires deeper document level understanding and transformer based pretrained language models, which are trained as masked language models, learn to represent tokens instead of representing the individual sentences (Liu and Lapata, 2019). In order to overcome this inability of pre-trained masked language models for summarization tasks, each sentence in the article should be represented not only individually but also, should contain the semantic information from the other sentences in the same article. Therefore, input data must be manipulated with some extra preparation as detailed below.

First, each article tokenized into sentences with Python nltk library. Then, [CLS] and [SEP] tokens were added at the beginning and end of the sentences in the original article, respectively. So, BERTurk and ELEC**TR**A language models can detect the sentence boundaries with the aim that each [CLS] token can collect and absorb important features for the sentence it represents. After that, by using BERTurk base tokenizer and ELEC**TR**A base tokenizer, the articles and the summaries are tokenized to reach the input indexes of each token based on the related tokenizer. Since both BERTurk and ELEC**TR**A language models were trained with 512 tokens, sub word level truncation and padding operations were applied to each article so that each has 512 tokens. Finally, the attention masks are added in order to indicate padding tokens and avoid performing attention on them. In other words, masking the padded inputs helps the self-attention mechanism to attend only required information.

Since the summarization task requires to distinguish multiple sentences in the article, interval segment embeddings were used for that purpose. 0 was assigned to the tokens in i-th sentence, if i is even and 1 was assigned to the tokens in i-th sentence, if i is odd. With this way, article representations can be learned hierarchically (Liu and Lapata, 2019).

Consequently, token indexes of the original articles, CLS (starting) positions of the sentences, sentence labels, interval segment embeddings and the token indexes of the gold summaries are obtained for the sentence encoders, BERTurk and ELEC**TR**A separately. The contextual sentence representations for each article can be obtained by fine tuning pretrained language model encoders with this dataset and then used to perform sentence scoring and selection for extractive summaries.

3.4 Extractive summarization models

Extractive summarization intends to choose the most salient and significant sentences from the original input article in order to generate a summary of it. Treating the extractive summarization as a sequence labelling problem, a binary decision is required to be made for each sentence in the main article by assigning scores to them. Previously, this approach proves its effectiveness and achieves high ROUGE scores (Cheng and Lapata, 2016; Nallapati et al., 2016a; Liu and Lapata, 2019).

The selection of model architecture yields differences in the model performances. For example, Cheng and Lapata (2016) and Nallapati et al. (2016a) utilized LSTM layers to score sentences, but they could not pass the performance of Lead-3 baseline with their models. Although LSTMs are quite useful and effective in several NLP tasks, since they cannot be run in parallel, using them leads to high memory

usage and longer training time. In addition, text summarization models need to be trained with relatively longer textual data, unlike other common NLP tasks such as sentiment analysis, named-entity-recognition (NER) and part-of-speech (POS) tagging. Therefore, text summarization models need to capture and learn longer term dependencies than the LSTMs can achieve. Unlike LSTMs, Transformer networks can work in parallel and handle the longer term dependencies with ease. For example, best performing extractive summarization models proposed in the literature using the CNN/Daily Mail utilizes Transformer based pre-trained language models in their studies (Bae et al., 2019; Zhang et al., 2019; Zhong et al., 2019; Liu and Lapata, 2019; Zhong et al., 2020).

Based on these findings, the extractive summarization is treated as a sequence labelling process also in this thesis and different Transformer architectures are considered to constitute the investigated model architectures. Similar to the previously proposed approaches, pre-trained language models are used as the encoder that encodes sentences. On top of the language models, extra inter-sentence Transformer layers are put. These layers get the sentence representations as input and generate the contextual sentence embeddings with the aim of capturing document level features for extracting summaries. Finally, these contextual sentence embeddings are fed into the sigmoid layer where the sigmoid function takes the contextual embeddings and outputs a value between 0 and 1 (Equation 3.1) as a prediction for each sentence. These scores represent the probability that the related sentence is in the generated summary. Then, the predicted scores are compared with the original extracted labels. The summarization models then try to minimize the difference between the predicted scores and original labels utilizing the cross-entropy loss (Equation 2.5) by updating the model parameters.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

The model architecture that encodes the sentences using the last layers' related CLS tokens of the underlying language model, is shown in Figure 11. Each sentence in the main articles starts with [CLS] token and it ends with [SEP] token as mentioned in Section 3.3. Since each main article should be represented with fixed size length, each main article is represented with 512 tokens due to positional embeddings limitation of language models, and similarly there are 32 sentences in each main article as a result of sentence padding and truncation processes. At the bottom of proposed model's architecture, these sentences, which are tokenized to subword level with the related pre-trained language model's tokenizer, are fed into the language model as input. In the experiments, model weights of BERTurk base (32K), BERTurk base (128K) and ELEC**TR**A models were utilized as the encoder part of the proposed summarization model architectures. Hence, the pre-trained language model part of the architecture has the same 12 Transformer layers that exist in BERTurk and ELEC**TR**A models. This part outputs each token representation for each sentence in the main articles. After obtaining the token representations, the [CLS] tokens are chosen as related sentences' representations and they are passed into extra inter-sentence Transformer layers to extract document level features. Finally, the sigmoid layer takes extra Transformer layers' outputs and assigns a score between 0 and 1 to each sentence.

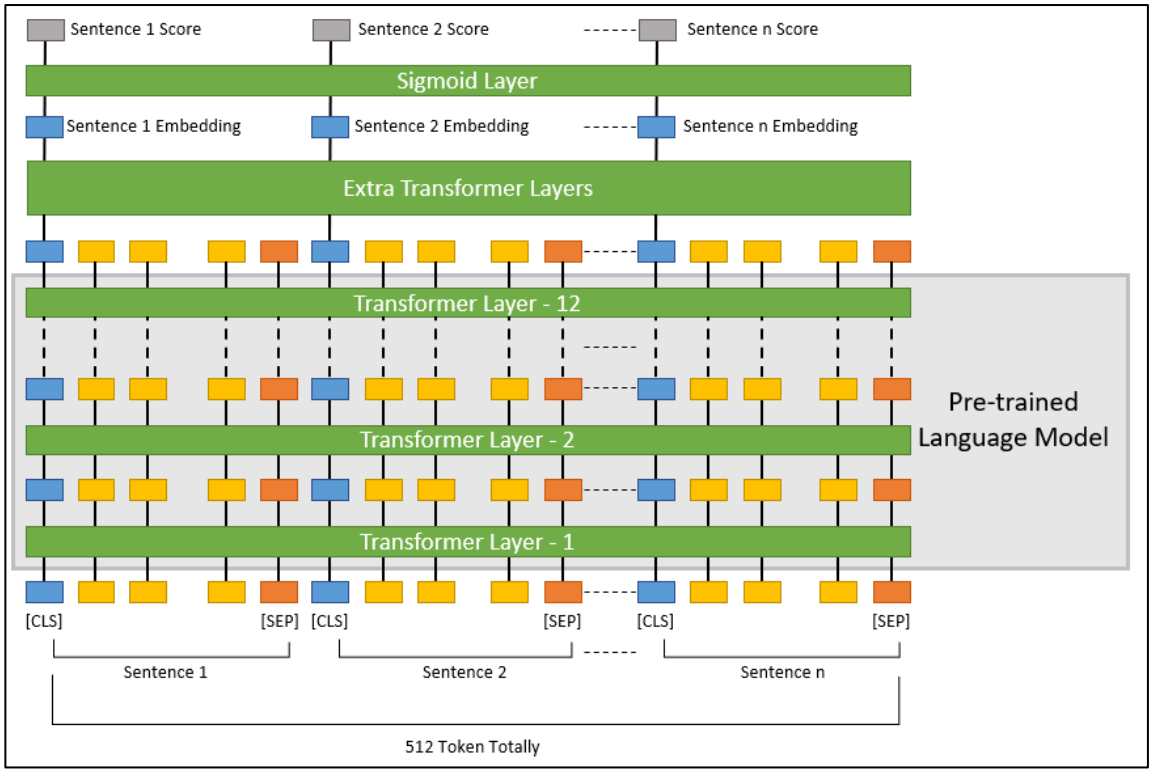

Figure 11. Architecture which uses CLS token representations to represent sentences

The highest scorer sentences, where the scores are obtained from the proposed summarization model, produce the predicted summary. Then, ROUGE score between the predicted and human-written summary is computed and used to evaluate the performance of the model. In this approach, it is assumed that the selected sentences represent the most important content of the main article. Besides, during the sentence selection process, Trigram Blocking is used as a post-processing step to reduce redundancies in the predicted summary (Paulus et al., 2018). In this heuristic approach, the next (candidate) sentence with the highest score is appended to the predicted summary if and only if trigram overlapping does not exist between the candidate sentence and previously selected sentences. The main reason for applying Trigram Blocking is to minimize the similarity between the sentences which have been already

selected as a part of the predicted summary and the next highest scoring candidate sentence. It can be seen as a control mechanism in order not to generate a summary which has repetitive content.

Besides the model architecture described above which uses CLS token representations for sentence encoding, some modifications over this architecture are investigated for Turkish news extractive summarization. It was mentioned that, as the underlying pre-trained language model, both BERTurk models and ELEC**TR**A models were utilized. But, instead of using the last layer's CLS tokens of each sentence, it is also possible to represent a sentence by taking the average of the last layer's token representations for the tokens that make up that sentence. This operation is called mean pooling. For example, Zhong et al. (2019) preferred to use *mean pooling* rather than CLS representations to get the sentence representations for the extractive summarization. Also, Reimers and Gurevych (2019) showed that using mean pooling may slightly improve the performance of text classification models compared to using CLS token representations. In this thesis, the use of mean pooling was also investigated where the modified model architecture based on mean pooling can be seen in Figure 12.

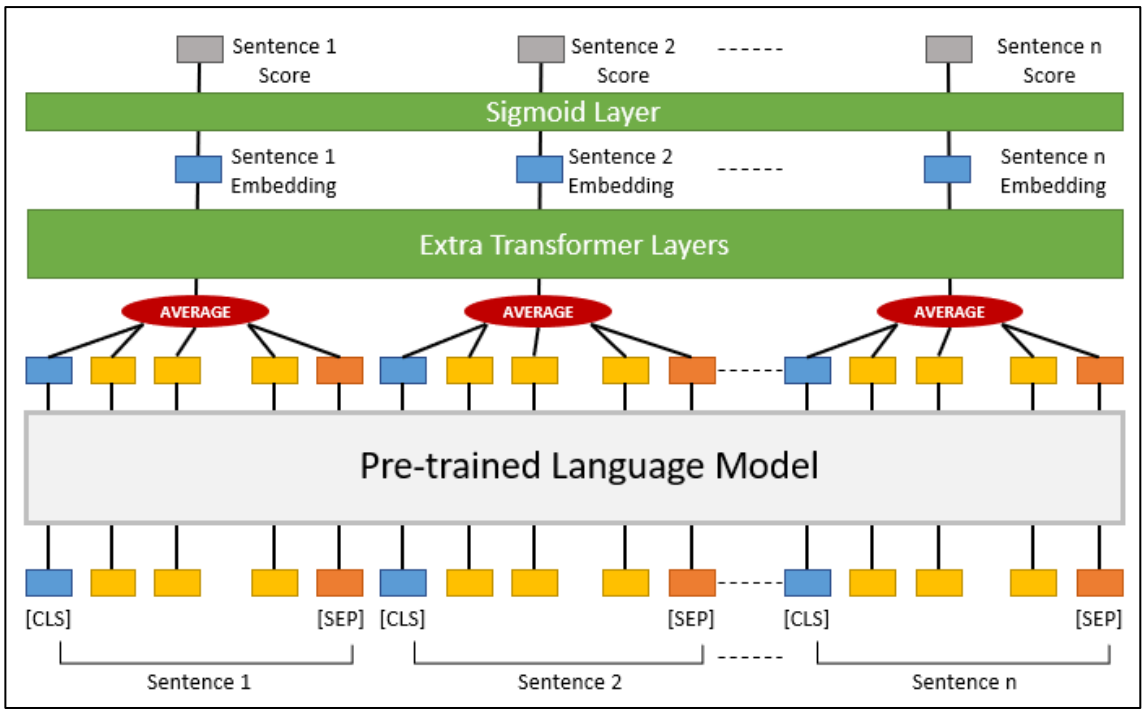

Figure 12. Architecture which uses mean pooling to represent sentences

In addition to using CLS and mean pooling representations, there are different approaches in the literature to get sentence representations from BERT-like language model architectures. For example, Devlin et al. (2018) stated that summing the CLS values of the last 4 layers can produce comparable performance results for NER tasks. Also, Reimers and Gurevych (2019) explained that concatenating a vector *u*, with a second one *v* and their absolute element wise differences |*u*-*v*|, may also be a good alternative to represent sentences. However, it should be considered that the dimension of sentence representations created in this way is three times larger than the previous ones as a result of this concatenation (*u*, *v*, |*u*-*v*|). Another alternative investigated in the same study is to concatenate directly two vectors (*u*, *v*). According to the study, this process makes the sentence vector dimensions doubled but did not produce comparable performance results in text classification tasks. In this study, each one of these sentence

representation approaches were investigated to compare their performances in an extractive summarization task. Beyond these previously investigated approaches, a new sentence representation approach is proposed in this study in which the CLS token representation of the last layer in the pre-trained language model is summed up with the mean pooling of the word representations. Thus, dimensions of the sentence representations stay the same as CLS token representation. The modified model architecture implementing this approach is shown in Figure 13.

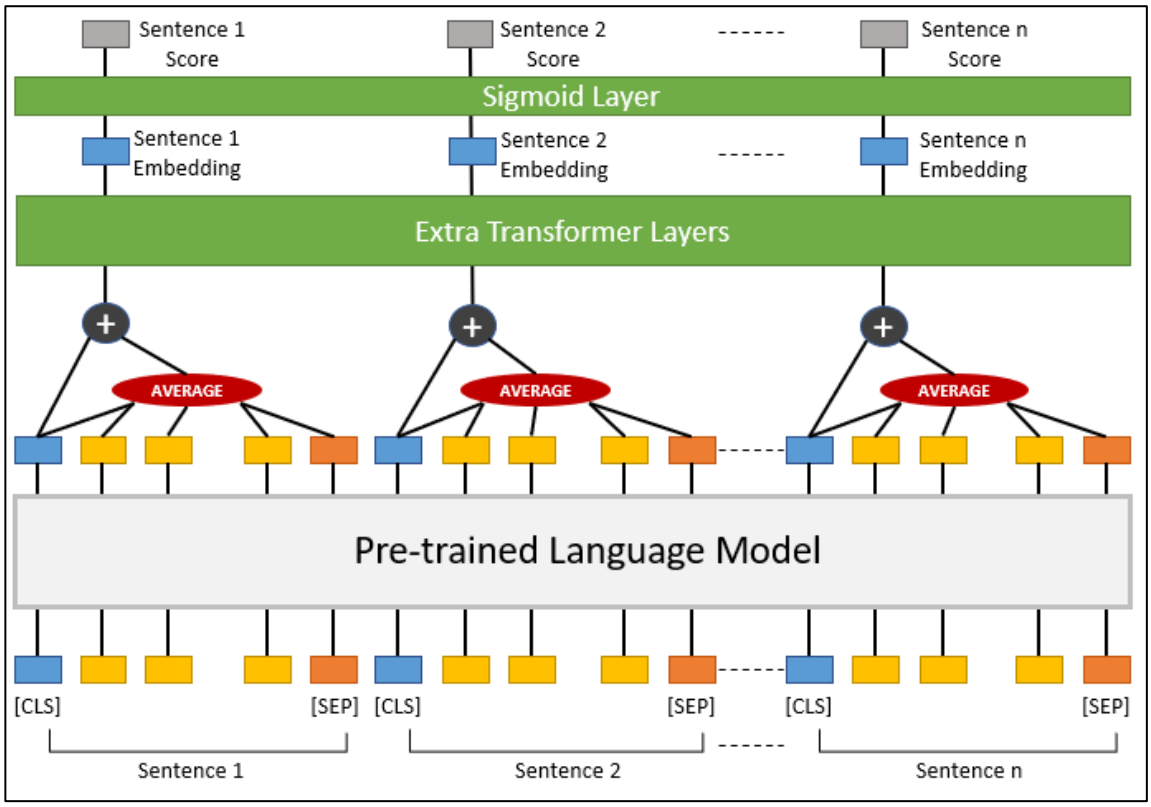

Figure 13. Architecture which uses the sum of CLS token and mean pooling to represent sentences

In the previously mentioned architectures, sentences are contextualized in the extra Transformer layers that are located above the base language model architecture. But, Guo et al. (2020) stated that the large networks like Transformers on top of BERT-

like language models does not improve the performance results by a large margin and they applied simple linear layer and sigmoid layer in their extractive summarization model architecture. In this study, in addition to analyzing the use of complicated Transformers layers, this simple architecture is also investigated to see if simplicity helps to improve the performances of the extractive summarization models where this simple architecture is shown in Figure 14.

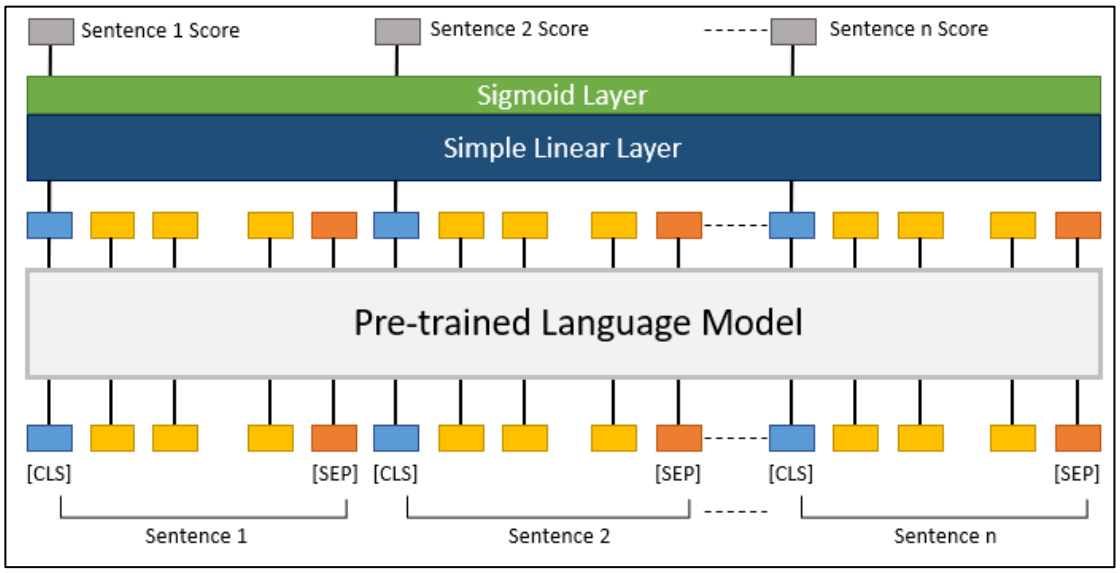

Figure 14. Architecture with a simple linear layer

In the next section, the details of the experiments with different architectures and hyperparameter settings are given and the performance results based on ROUGE scores are compared.

CHAPTER 4

EXPERIMENTS AND RESULTS

Different deep learning architectures considered in the thesis proposed for extractive summarization are introduced in the previous chapters. In this chapter, the experiments conducted by creating models with these different architectures are explained in detailed with their related architectures, hyperparameter settings and the performance results obtained by running these models over the Turkish news dataset are discussed.

For all experiments, one Tesla V100 16GB GPU was used and these experiments were implemented with PyTorch library. The batch size for each experiment was chosen as 8 due to the memory limitations. As the optimizer, ADAM optimization algorithm (Kingma and Ba, 2014) is deployed with the aim of updating the model parameters iteratively in the training data. The authors suggest that $\beta 1$ and $\beta 2$ hyperparameters of Adam optimizer work best with the values of 0.9 and 0.999, respectively. As in Devlin et al. (2018), Liu and Lapata (2019) and Clark et al. (2020), these values were also used in the experiments. Additionally, to schedule a learning rate, a linear scheduler with warm-up steps was deployed as it was done in Devlin et al. (2018) and Clark et al. (2020). In this type of scheduler, the learning rate increases linearly from zero to the initial given learning rate during the warm-up period and after that, it decreases linearly from that learning rate to zero again. Finally, the dropout probabilities are kept at 0.1 for each experiment (Devlin et al., 2018). To sum up, common settings that were used in each experiment is shown at Table 3.

Table 3. Common Settings for each Experiment

Optimizer:	Adam
Adam β1:	0.9
Adam β2:	0.999
Scheduler:	Linear
Dropout Probabilities:	0.1

4.1 Hyperparameter selection

As stated in Section 3.4, different model architectures were proposed and in this section the effects of different values of hyperparameters and choices of underlying deep learning architectures are investigated.

Firstly, the effects of pre-trained language models on the performance results are investigated. In this context, three different language models were considered. These were ELEC**TR**A, BERTurk base (32K) model (which has 32 thousand words in its vocabulary) and BERTurk base (128K) model (which has 128 thousand words in its vocabulary). Even though their architectures are quite similar, both architectures have 12 layers Transformers network with the hidden size 768, ELEC**TR**A and BERT models differ from each other in terms of their objective functions. ELEC**TR**A uses RTD, whereas BERT uses MLM. Considering these model in this thesis provides us with the effect of using models of similar underlying architecture trained with different objective functions. Besides, comparison of the performance results of BERTurk base (32K) and BERTurk base (128K) models are expected to provide insights on the effect of utilizing large models trained with larger vocabulary.

Secondly, the effect of increasing the capacity of the models by putting extra Transformer layers' was examined by checking the ability of the models to capture document level features in the process of learning sentence representations (Figure 11, 12 and 13) by comparing their performances with performance of the model that has single linear layer (Figure 14). Beyond that, different number of extra Transformer layers, 1, 2 and 3 layers, was implemented and the performances were analyzed despite the fact that Liu and Lapata (2019) found out that the best performance is achieved with 2 extra Transformer layers.

Thirdly, the different sentence representation approaches were experimented as explained in Section 3.4. These different sentence representations used to select those sentences to be included in the extractive summary were obtained by extracting the CLS token of the pre-trained language model's last layer, applying mean pooling over the token representations of the last layer, summing up CLS token and mean pooling representations, concatenating CLS token and mean pooling representations, concatenating CLS token, mean pooling representations and the absolute value of their element wise differences, and finally, summing up CLS token representations of the pre-trained language model's last 4 layers. The list of these architectural settings and their different values are shown in Table 4.

Table 4. Architectural Settings Used in the Experiments

Pre-trained Language Models:	ELEC**TR**A		BERTurk base (32K)		BERTurk base (128K)	
Extra Layers:	Single Linear Layer		1-layer Transformer	2-layer Transformer	3-layer Transformer	
Sentence Representation Approaches**:**	CLS token of last layer	Mean pooling over last layer	Summed CLS token with mean pooling	Concatenating CLS token and mean pooling	Concatenating CLS token, mean pooling and their absolute differences	Summing up CLS token of last 4 layers

In addition to architectural settings, suitable training hyperparameters were also investigated. These hyperparameters include the number of training epochs, learning rate and feed forward network's (FFN) hidden size and number of attention heads in the extra Transformer layers. The experimented values for each hyperparameter are shown in Table 5.

Table 5. Training Hyperparameter Sets Used in the Experiments

# of Epochs	Learning Rate	FFN Hidden Size	# of Attention Heads
4	5E-5	2048	8
		3072	12
5	1E-4	4096	16
		6144	24
10	2E-3	9216	32

The epoch indicates the number of passes of feeding whole training data in mini-batches to the deep learning models. In the experiments, the values considered for the number of training epochs are 4, 5 and 10. For extra Transformer layer-based architectures, models were fine-tuned for 4 and 5 epochs. For the simple linear layer-based architectures, the models were trained for 10 epochs. The main reason for this difference is the fact that the Transformer networks have much more parameters and are more capable of learning from the training data, as compared to shallow networks like linear layers. Therefore, to increase the learning capability of linear layers, more training epochs were used.

The second training hyperparameter is the learning rate. The learning rate can be defined as the step size that the model uses when updating its parameters. It controls the speed at which the model learns and has direct effect on the convergence properties of the models. Devlin et al. (2018) recommends 5e-5 for the learning rate in the fine-tuning processes, whereas Clark et al. (2020) recommends 1e-4. Additionally, Liu and Lapata (2019) used 2e-3 as the learning rate in their extractive summarization models. In the experiments conducted in this thesis, each of these learning rates were applied in order to explore their effects on the model performances.

The third training hyperparameter size is the FFNs' hidden size and number of attention heads in the extra Transformer layers as previously shown in Table 5. Liu and Lapata (2019) used 2048 and 8 for these hyperparameters, respectively. However, Devlin et al. (2018) set the FFN hidden unit size to be 4 times the hidden units in the Transformer architecture in all of their experiments. Since the number of hidden units in both BERT and ELECTRA equal to 768, FFN hidden size was set to 3072 in this thesis where the sentence representations are of size 768. For the concatenation-based sentence

representation approaches (Table 4), the number of hidden units increases, therefore, the FFN hidden sizes are increased with the same proportion. For example, CLS token representations, mean pooling representations and summing CLS token with mean pooling approaches have 768 hidden units and, in these settings,, FFN hidden size were set to 2048 and 3072, with 8 and 12 attention heads, respectively. However, concatenating CLS token representations with mean pooling representations leads to 1536 (768+768) hidden units and as a result, FFN hidden sizes were set to 4096 and 6144 with 16 and 24 attention heads, respectively. Additionally, the ratio of FFN hidden size to number of attention heads is kept constant at 256 for all experiments. For simple linear layer-based experiments, these hyperparameters are not applicable. The overall hyperparameter settings used in different sentence representation approaches is shown in Table 6.

Table 6. Experimented FFN Hidden Size and Number of Attention Heads based on Sentence Representation Approaches

Sentence Representation Approach	Hidden Unit Size	FFN Hidden Size and # of Attention Heads
CLS	768	(2048,8), (3072,12)
Mean pooling	768	(2048,8), (3072,12)
CLS + Mean pooling	768	(2048,8), (3072,12)
Summing CLS tokens of last 4 layers	768	(2048,8), (3072,12)
Concatenating CLS and Mean pooling	1536	(2048, 8), (3072, 12), (4096, 16), (6144, 24)
Concatenating CLS, Mean pooling and |CLS-mean pooling|	2304	(2048, 8), (3072, 12), (6144, 24), (9216, 36)

4.2 Experimental details

As it is explained in the previous section, in total there were 7 different hyperparameter settings to be optimized. Three of them were architectural, namely different underlying language models, number of extra layers, and sentence representations. Remaining four were training hyperparameters, namely different FFN hidden sizes, number of attention heads, learning rates and number of epochs. Since experimenting with all of the combinations of these hyperparameters is time and resource consuming, some of the settings were filtered out based on the performances obtained from experimenting with different pre-trained language models and sentence representation approaches. As a result, 241 different models were trained with different hyperparameter settings. The architectures and hyperparameter settings for each of these models can be seen in Appendix A together with their performance results and training times.

Firstly, 78 models were trained with CLS token representations and same number of model were trained with mean pooling representations. The applied hyperparameter values for these 2 sentence representation approaches and the number of models run can be seen in Table 7.

Table 7. Hyperparameter Combinations for Models Trained with CLS Token and Mean Pooling Representations

	Language Model	Extra Layers	FFN Hidden size with # of Attention Heads	Epoch	Learning Rate	Total # of Models
CLS Token	BERTurk base (32K), ELEC**TR**A	Simple Linear Layer	-	10	5e-5, 1e-4, 2e-3	6
	BERTurk base (32K), ELEC**TR**A	1, 2 and 3 Transformer Layers	2048-8, 3072-12	4, 5	5e-5, 1e-4, 2e-3	72
Mean Pooling	BERTurk base (32K), ELEC**TR**A	Simple Linear Layer	-	10	5e-5, 1e-4, 2e-3	6
	BERTurk base (32K), ELEC**TR**A	1, 2 and 3 Transformer Layers	2048-8, 3072-12	4, 5	5e-5, 1e-4, 2e-3	72

After running these 156 models, the top-10 performing settings were selected based on ROUGE-2 F scores, as seen in Appendix B. The main reason behind the selection of ROUGE-2 score for model comparison is the fact that the extractive labels were obtained with a greedy algorithm (Nallapati et al., 2016a) where the ROUGE-2 scores of selected sentence sets were maximized with respect to human written summaries. Then, hyperparameter setting of these top 10 performing models were used for the other four sentence representation approaches. In addition to these extra models, for the concatenation based sentence representations, FFN hidden sizes and number of attention values were multiplied by the number of representations concatenated in the model as shown in Table 6. Total number of different models for each sentence representation approach is shown in Table 8.

Table 8. Total Number of Different Models Trained by Implementing each Sentence Representation Approach

Sentence Representation Approach	# of Different Models in the Experiments
CLS	78
Mean pooling	78
CLS + Mean pooling	10
Summing CLS tokens of last 4 layers	10
Concatenating CLS and Mean pooling	20
Concatenating CLS, Mean pooling and |CLS-mean pooling|	20
TOTAL	216

To see the effect of vocabulary size of the pre-trained language models on the model performances, hyperparameters of the top-25 performing models (which are all BERTurk base (32K) models) among these 216 models were selected based on ROUGE-2 F score (see Appendix C). Then, keeping the other hyperparameter settings the same, the pre-trained language model hyperparameter was replaced with the BERTurk base (128K) model. In the end, in total 241 models were trained with different hyperparameter settings and evaluated.

4.3 Performance results

In the experiments, ROUGE scores (Lin, 2004) between human written summaries and automatically generated summaries by the models were computed in order to evaluate the *fluency* with ROUGE-1 and ROUGE-2 scores and assess *informativeness* with ROUGE-L score. In Appendix A, all 241 models are reported with their ROUGE-1 F scores, ROUGE-2 F scores and ROUGE-L F scores over the test data together with the training time of each model.

Before comparing and interpreting these results, the upper bound (ORACLE) and the baseline (LEAD-5) performance scores were calculated. ORACLE approach does not contain any training part. The sentences selected by the greedy algorithm (Nallapati et al., 2016a), were considered as the final summaries. The score obtained by these sentences is considered as the maximum score one could achieve in extractive summarization (Scialom et al., 2020). Therefore, the ORACLE ROUGE scores were chosen as the upper bound. On the other hand, LEAD-5 simply selects the first five sentences in the input document to generate final summaries and it is presented as a baseline for the other methods and the score obtained from the LEAD-5 was considered as the lower bound. As stated in Section 3.2, LEAD-5 is a strong baseline for Turkish news dataset since the distribution of extracted sentence positions is not uniform and these important sentences are mostly placed at the beginning of main articles. The ROUGE scores of these models on test data are shown in Table 9.

Table 9. The ROUGE Scores of ORACLE and LEAD-5

	ROUGE-1 F Score	ROUGE-2 F Score	ROUGE-L F Score
ORACLE	53.60	41.63	52.80
LEAD-5	37.49	26.40	37.12

Among all 241 models trained in this study, 33 of them perform better than LEAD-5 ROUGE F scores as shown in Table 10. As stated in the Introduction chapter, the results are interpreted in terms of utilizing different pre-trained language models, sentence representation approaches, and architectural simplicity and complexity.

4.3.1 Effects of pre-trained language models

In the experiments, three different pre-trained language models were utilized. These are BERTurk base (32K), BERTurk base (128K) and ELEC**TR**A. As mentioned in Section 4.2, the experiments were conducted first with 78 different parameter settings using ELEC**TR**A as the underlying pre-trained language model. The hyperparameter settings and ROUGE scores are reported in Appendix A. It can be seen that a major portion of the 78 models obtained the same ROUGE F scores with LEAD-5 baseline. These experiments put extra Transformer layers on top of the ELEC**TR**A. Therefore, it can be concluded that these models learned only positional embeddings of the sentences in the main articles and generate final summaries by considering their positions and extracting the first 5 sentences. The possible reasons for this might be the size of the dataset used to fine-tune the models. In the training set, there were 1476 articles, and this may not be sufficient to fine tune the ELEC**TR**A model and hence for the model to learn semantic and contextual relationships between the sentences.

Table 10. The Performance Results of Best 33 Models which are Better than LEAD-5 Baseline with Their Corresponding Settings

#	Language Model	Sentence Representation	Extra Layer	FFN Size	# of Attention Heads	# of Epoch	Learning Rate	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
160	BERTurk-Base (32K)	CLS Token + Mean pooling	2-Layer	2048	8	5	1.00E-04	38.38	26.8	38.04
206	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer	6144	24	5	5.00E-05	37.85	26.59	37.53
184	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer	4096	16	4	1.00E-04	37.83	26.57	37.47
175	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer	2048	8	4	5.00E-05	37.75	26.56	37.38
164	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer	2048	8	4	1.00E-04	38.03	26.55	37.66
174	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer	2048	8	4	1.00E-04	37.7	26.54	37.36
28	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer	3072	12	5	5.00E-05	37.8	26.53	37.45
17	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer	3072	12	4	1.00E-04	37.78	26.5	37.43
166	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer	2048	8	5	5.00E-05	38.03	26.5	37.6
205	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer	6144	24	4	5.00E-05	37.71	26.5	37.34
11	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer	3072	12	4	1.00E-04	37.72	26.49	37.36
29	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer	3072	12	5	1.00E-04	37.71	26.49	37.37
159	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer	2048	8	5	1.00E-04	38.02	26.49	37.62
172	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer	3072	12	4	1.00E-04	37.73	26.48	37.35
32	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer	2048	8	5	1.00E-04	37.8	26.47	37.41
26	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer	2048	8	5	1.00E-04	37.69	26.46	37.38
158	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer	3072	12	4	1.00E-04	37.91	26.46	37.5
14	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer	2048	8	4	1.00E-04	37.66	26.45	37.3
168	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer	3072	12	4	1.00E-04	37.72	26.45	37.32
182	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer	6144	24	4	1.00E-04	37.69	26.45	37.34
196	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer	2048	8	5	5.00E-05	37.74	26.45	37.39
13	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer	2048	8	4	5.00E-05	37.62	26.44	37.27
163	BERTurk-Base (32K)	CLS Token + Mean pooling	2-Layer	2048	8	5	5.00E-05	37.98	26.44	37.59
25	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer	2048	8	5	5.00E-05	37.69	26.43	37.33
31	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer	2048	8	5	5.00E-05	37.68	26.43	37.31
7	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer	2048	8	4	5.00E-05	37.63	26.42	37.28
165	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer	2048	8	4	5.00E-05	37.88	26.42	37.49
217	BERTurk-Base (128K)	CLS Token + Mean pooling	2-Layer	2048	8	5	1.00E-04	37.83	26.42	37.51
5	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer	3072	12	4	1.00E-04	37.71	26.41	37.36
8	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer	2048	8	4	1.00E-04	37.64	26.41	37.32
191	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	2-Layer	3072	12	5	1.00E-04	37.53	26.41	37.18
195	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer	2048	8	4	5.00E-05	37.56	26.41	37.2
199	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer	6144	24	5	1.00E-04	37.67	26.41	37.32

Unlike models that use ELEC**TR**A as the underlying language model, models that use BERTurk base (32K) language model performed quite well. As seen in Table 10, among the 33 best performing models, 32 of them use BERTurk base (32K) language model. Moreover, in the first part of experimental design, 78 different settings were chosen for ELEC**TR**A and BERTurk base (32K) language models, separately. In the second part, top 10 performing models were chosen among these 156 models to investigate the effects of using other sentence representation techniques. Performance results and hyperparameter settings of these top 10 best performing models are shown in Appendix B. All of them have BERTurk base (32K) as their underlying pre-trained language model. Therefore, it can be said that the BERTurk base (32K) model performs better than ELEC**TR**A for Turkish news extractive summarization.

Finally, BERTurk base (128K) pre-trained language model is used as the pre-trained language model for the models trained with the hyperparameter settings of the top 25 best performing BERTurk base (32K) models which are shown in Appendix C. The performances obtained by these 25 models are not as good as the ones obtained by the models utilizing BERTurk base (32K) as the underlying language model, as seen in Appendix A. As it can be seen from Table 10, best performing BERTurk base (128K) based model takes 28th place over all the model performances. Its settings are the same with the best model in the Table 10 except for the difference in the pre-trained language model. As a result, it can be said that generating a model using BERTurk base (32K) is better than utilizing BERTurk base (128K) with approximately two thousand training examples for the extractive summarization task of Turkish news. The reason why smaller vocabulary performs better than larger one might be that 32K may be good enough to represent all the words in the small training data. With 128K vocabulary and

small dataset, each token is represented lesser and the model may not learn their embeddings properly in the fine-tuning phase.

The average ROUGE F scores of the models utilizing different pre-trained language models in all 241 experiments conducted in this thesis are shown in Table 11. The reason why the average ROUGE F scores of the models with different pre-trained language models are compared among all 241 models is that there are not enough ELECTRA and BERTurk base (128K) language models in the best performing models for comparison properly. As it is seen from the Table 11, utilizing BERTurk base (32K) as the underlying language model is better than using BERTurk base (128K) and ELEC**TR**A.

Table 11. The Average ROUGE Scores of all 241 Models Generated by Utilizing Different Pre-trained Language Models

Language Model	# of models	ROUGE-1 F Score	ROUGE-2 F Score	ROUGE-L F Score
BERTurk base (32K)	138	37.06	25.72	36.70
BERTurk base (128K)	25	36.75	25.52	36.35
ELEC**TR**A	78	35.26	24.04	34.83

As shown in Table 11, the number of models using these pre-trained language models are not close to each other. Hence, the comparison with respect to these scores may not yield strong evidence. In order to more conveniently compare the effects of underlying language models on extractive summarization models, the 30 models which have exactly the same architectures and hyperparameters sets except the pre-trained language model parts are compared in Table 12. Based on the findings from these models, the models utilizing BERTurk base (32K) performs better than the models using

BERTurk base (128K) and ELEC**TR**A. However, there is no significant difference between performance results of BERTurk base (128K) and ELECTRA based models.

Table 12. The Average ROUGE Scores of the Models Generated by Utilizing the Same Settings Except the Pre-trained Language Models

Language Model	# of models	ROUGE-1 F Score	ROUGE-2 F Score	ROUGE-L F Score
BERTurk base (32K)	10	37.72	26.47	37.36
BERTurk base (128K)	10	37.49	26.40	37.12
ELEC**TR**A	10	37.49	26.40	37.12

4.3.2 Effects of architectural simplicity/complexity

In order to investigate the performance gained obtained by the capacity of the models, four different extra layers that are put on top of the underlying pre-trained language models are considered in this thesis. These extra layers are simple linear layer, 1-layer Transformer network, 2-layer Transformer network and 3-layer Transformer network. The models obtained by fine-tuning pre-trained models with a simple linear layer performed worse compared to the models formed using Transformer layers. Guo et al. (2020) stated that the complex networks on top of BERT do not lead to a large margin performance gains in extractive summarization tasks, and they applied simple linear layer as extra layer. As opposed to their findings, simple linear layer based models in this study produced lower ROUGE scores, and they are even worse than LEAD-5 baseline. Similarly, the model implemented with 1-layer Transformer network did not perform well as compared to the ones having 2 and 3 Transformer layers. Finally, performances obtained from the models with 3-layer Transformer networks are slightly

better than the ones with 2-layer Transformer network with respect to average ROUGE F scores. These can be observed from the results given in Table 10.

As a result, more complex models are found to produce higher performance scores in extractive summarization task for Turkish news. The average ROUGE F scores of best 33 models using different extra Transformer networks are shown in Table 13. In addition to average scores, maximum and minimum scores are also reported since the number of models using these extra layers are very different.

Table 13. The Average, Maximum and Minimum ROUGE Scores of Best 33 Models Generated by Utilizing Different Extra Layers

Extra Layer	# of models	ROUGE-1 F Score (Avg/Max/Min)	ROUGE-2 F Score (Avg/Max/Min)	ROUGE-L F Score (Avg/Max/Min)
1-Layer Transformer	1	37.71/37.71/37.71	26.41/26.41/26.41	37.36/37.36/37.36
2-Layer Transformers	13	37.77/38.38/37.53	26.48/26.80/26.41	37.42/38.04/37.18
3-Layer Transformers	19	37.79/38.03/37.56	26.48/26.59/26.41	37.42/37.66/37.20

4.3.3 Effects of sentence representation approaches

In these approaches, sentences are represented with the last layer's CLS token representation of the pre-trained language models, mean pooling of the token representations over the last layer, summing up CLS token and mean pooling representations, concatenating CLS token and mean pooling representations, concatenating CLS token, mean pooling representations and the absolute value of their element wise differences, and finally, summing up CLS token representations of the pre-trained language model's last 4 layers. In the literature, CLS token representations and mean pooling over the last layer's token representations are the most common

approaches. Although Reimers and Gurevych (2019) found out that mean pooling can result in better performance scores compared to CLS token representations, the models using CLS token representations produced better average ROUGE F scores in the 241 experiments conducted in this study, as shown in Table 14. The main reason for the mean pooling performing worse might be the fact that, with mean pooling, each token is averaged with the same weight, including stop words or other tokens that are not significant for the summaries. The CLS token representations are computed using self-attention, so it can only collect the relevant information from the rest of the hidden states. Therefore, combining both CLS and mean pooling may lead to good results. These representations can be combined by either concatenating them or by doing element-wise sum. Reimers and Gurevych (2019) have tried several concatenation approaches and found out that concatenating CLS, mean pooling and absolute value of their element wise differences produces best performance results. Moreover, they stated that concatenating CLS and mean pooling directly may not perform well. Parallel to that, concatenating CLS and mean pooling directly is the second worst performing approach according to the averages obtained from the models generated by this approach, even though among these 26 models, 6 of them were able to beat LEAD-5 baseline. On the other hand, approaches that employ the approach that sum the token representations achieved the best average scores. Devlin et al. (2018) stated that summing the last four layers' CLS tokens is a good sentence representation technique. As it can be seen in Table 14, the models using this technique obtained the second-best average ROUGE scores. However, none of the models generated with this approach exceed LEAD-5 ROUGE-2 F score. None of the models implemented this approach takes place in the best 33 ones which are the ones that exceed baseline LEAD-5 score as can be seen in

Table 10. Finally, a new representation approach which is summing CLS representation with mean pooling is proposed. This approach is the best one among the others. The reason might be that the final sentence representation takes both CLS token and mean pooling representations into account, so that both stop words and irrelevant tokens are represented but their worsening effect might be mitigated on final representation by the CLS token as compared to the mean pooling. Besides, the tokens that are far away from the CLS token in the sentence can be better represented this way as compared to using only CLS tokens.

Table 14. The Average ROUGE Scores of all 241 Models Generated by Utilizing Different Sentence Representation Approaches

Sentence Representation Approach	# of models	ROUGE-1 F Score	ROUGE-2 F Score	ROUGE-L F Score
(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	23	37.11	25.89	36.75
(CLS Token, Mean pooling)	26	36.38	25.09	35.99
CLS Token + Mean pooling	16	37.85	26.39	37.50
CLS Token of the Last Layer	88	36.47	25.22	36.07
Mean pooling of the Last Layer	78	35.79	24.50	35.37
Sum Last Four Layers' CLS Token	10	37.68	26.19	37.35

To eliminate the effect of comparing average scores of the sentence representation approaches with different number of models and do more appropriate ablation study, the average ROUGE F scores of 60 models, which have the same architectures and hyperparameters, are compared in Table 15. According to these scores,

the models that are utilizing the summation of CLS token and mean pooling representations achieved the highest performance scores.

Table 15. The Average ROUGE Scores of the Models Generated by Utilizing the Same Settings Except the Sentence Representation Approaches

Sentence Representation Approach	# of models	ROUGE-1 F Score	ROUGE-2 F Score	ROUGE-L F Score
(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	10	36.51	25.29	36.13
(CLS Token, Mean pooling)	10	37.56	26.26	37.21
CLS Token + Mean pooling	10	37.95	26.47	37.58
CLS Token of the Last Layer	10	37.72	26.47	37.36
Mean pooling of the Last Layer	10	37.48	26.26	37.13
Sum Last Four Layers' CLS Token	10	37.68	26.19	37.35

Finally, the effect of FFN hidden size were investigated. As stated earlier, Devlin et al. (2018) suggested setting the FFN hidden unit size to 4 times size of the hidden units in the Transformer architecture. Since concatenation-based sentence representations increases size of the hidden units, different FFN hidden size were experimented. For example, concatenating CLS token and mean pooling representations lead to 1536 (768+768) dimensional sentence representations. Therefore, both 4096 and 6144 (1536*4) FFN hidden sizes were experimented for this approach, in addition to 2048 and 3072 FFN hidden sizes, as shown in Table 6. The average ROUGE F scores of the models trained with both original and increased FFN hidden units are shown in Table 16. The results displayed in italics obtained from the models trained with

increased FFN hidden units. For concatenation of CLS token, mean pooling and their element wise absolute difference, increasing FFN hidden size achieves higher performance scores than original ones, whereas increasing FFN hidden size leads to worse performance scores than original ones for concatenating CLS token and mean pooling representations. Therefore, these scores do not provide strong evidence that increasing the FFN hidden units based on the hidden units in the Transformer architecture leads to better performance results.

Table 16. The Average ROUGE Scores of the Models Generated by Utilizing the Same Settings Except the Hidden Sizes

Sentence Representation Approach	# of models	ROUGE-1 F Score	ROUGE-2 F Score	ROUGE-L F Score
(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	10	36.51	25.29	36.13
(CLS Token, Mean pooling)	10	37.56	26.26	37.21
(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	*10*	*37.59*	*26.35*	*37.23*
(CLS Token, Mean pooling)	*10*	*36.56*	*25.25*	*36.18*

As a result, summing CLS token and mean pooling representations of the last layer is turned out to be the best sentence representation technique for summarizing Turkish news dataset. The model generated by combining BERTurk base (32K) model with this approach and putting extra 2-layer Transformer network on top of it has achieved the best ROUGE F scores. The best performing model with this architecture was trained for 5 epochs where learning rate and FFN hidden size parameters are set to 1e-4 and 2048, respectively. Almost 1 percent higher score than LEAD-5 ROUGE-L F score was achieved with this configuration, and hence it is possible to produce meaningful extractive summaries.

CHAPTER 5

CONCLUSIONS AND MANAGERIAL IMPLICATIONS

This thesis aims to develop an automated extractive summarization system for Turkish news. For this purpose, the most salient and significant sentences in the main articles are determined and extracted by the proposed models to generate final summaries. To develop an end-to-end extractive summarization model, the dataset is gathered via a well-known news website. The dataset consists of the news published in Turkish and the related human written summaries. After 2076 such news-summary pairs are collected, the data is preprocessed. Then, binary sentence labels in the main articles are extracted by greedy algorithm (Nallapati et al., 2016a) since the human-written summaries should be converted to extractive summaries to train the model. In the label extraction phase, it is realized that the most significant sentences are placed at the beginning of the main articles to capture the readers' attention and there is no uniform distribution for extracted sentence positions as shown in Figure 9. After extracting the binary labels for each sentence in each article, the most promising Transformer based pre-trained language models generated for Turkish are determined. There are BERT (Devlin et al., 2018) and ELECTRA (Clark et al., 2020) based models since both have been pre-trained for Turkish with huge corpus by Schweter (2020). The main purpose of using these pre-trained language model in the models proposed in this thesis is their ability to capture high-level textual features, such as semantic relationship between the words. Three different pre-trained language models are considered in generating summarization models, and then their abilities in Turkish news extractive summarization task are investigated. These models are named as ELEC**TR**A, BERTurk base (32K) and

BERTurk base (128K). ELECTRA is the language model trained based on the replaced token detection, while BERT is a masked language model. The difference between the performance results of BERTurk base (32K) and ELEC**TR**A may show the effectiveness of language model types over Turkish news extractive summarization. Besides, both BERTurk base (32K) and BERTurk base (128K) have the same architectures but their vocabulary size in the pre-training phase is different. Therefore, it is informative to see the effect of vocabulary size on the performance results by employing these two models.

On top of pre-trained language models, extra layers are added to capture document level features like intersentential relationships. Since Guo et al. (2020) stated that the larger networks does not contribute a large margin to the performance results and added an extra simple linear layer on top of the BERT model, the effect of complexity of the extra layers on the summarization performance results for Turkish news is also investigated in this thesis. Liu and Lapata (2019) applied 1-layer, 2-layer and 3-layer Transformer networks on top of BERT, separately and found out that 2-layer ones work best. In this thesis, four different extra layer alternatives are investigated. These are simple linear layer, 1-layer, 2-layer and 3-layer Transformer networks. Examining the performance results of the models generated by these different extra layer alternatives may yield understanding of the effect of architectural complexity.

The other architectural settings investigated in this study is the sentence representation approaches. As stated earlier, the pre-trained language models output each token representation in the sentence and to obtain sentence embeddings, different approaches were proposed in the literature. For example, Liu and Lapata (2019) used the CLS token embeddings to represent sentences, whereas Reimers and Gurevych (2019) stated that taking average of token embeddings, called mean pooling, in the sentences

may lead to better performance compared to the CLS token. Additionally, Devlin et al. (2018) found out that summing the last 4 layers' CLS token representations is quite good to represent sentences. Moreover, Reimers and Gurevych (2019) suggested that concatenation operations may yield good performance results. Therefore, concatenating CLS token representations with mean pooling representations, and also, concatenation of CLS token, mean pooling and their element wise absolute difference are also experimented. Finally, directly summing up CLS token representations with mean pooling representations is proposed in this study as a sentence representation approach. As a result, six different approaches were examined to measure and compare their effectiveness.

In addition to these architectural settings which are shown in Table 4, some training hyperparameters were also optimized. These hyperparameters are learning rate, number of epochs for training, FFN hidden size and number of attention heads. The last two hyperparameters are valid for only extra Transformer layer networks. The value sets for each training hyperparameter were presented in Table 5.

In total, 241 models with different architectures and hyperparameter setting were experimented in this thesis. Model architectures and the hyperparameters used to train models together with the performance results can be seen in Appendix A. Among these models, 33 of them produced higher ROUGE scores than LEAD-5 baseline. Based on these 33 models' ROUGE scores, the most effective pre-trained language model for Turkish news is found to be BERTurk base (32K). Therefore, it can be concluded that the BERTurk base (32K) performed better than ELEC**TR**A and lower vocabulary size (BERTurk base (32K)) enables models to capture token representations more effective than larger vocabulary size (BERTurk base (128K)) for Turkish news.

Also, architectures like 2-layer or 3-layer Transformer networks with higher capacities perform better than simpler ones like simple linear layer or 1-layer Transformer networks. This means that the complex networks achieve better performance results than simpler ones in this thesis.

Finally, summing CLS token representations with mean pooling representations to represent sentences in summarization model turns out to be the best approach compared to other ones considered in this thesis which can be seen in Table 17. The proposed approach aggregates contextual token embeddings more suitable in order to represent sentences compared to other popular sentence representation approaches.

Table 17. Average ROUGE Scores of the Best 33 Models based on Applied Sentence Representation Approaches

Sentence Representation Approach	# of models	ROUGE-1 F Score	ROUGE-2 F Score	ROUGE-L F Score
(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	6	37.68	26.46	37.33
(CLS Token, Mean pooling)	6	37.74	26.51	37.37
CLS Token + Mean pooling	8	38.01	26.51	37.63
CLS Token of the Last Layer	13	37.70	26.46	37.35

The best model among the 241 models achieves 38.38 ROUGE-1 F score, 26.8 ROUGE-2 F score and 38.04 ROUGE-L F score. These scores are significantly greater than the LEAD-5 baseline score which is a really strong baseline since significant sentences are mostly located at the beginning of the main articles to impress the readers.

Therefore, the results obtained in this study are promising. The best model uses BERTurk base (32K) as the underlying pre-trained language model, and sum of CLS token and mean pooling as the sentence representation approach, and put 2 extra Transformer network layers. Besides, hyperparameter values for FFN hidden size, attention heads, and learning rate are set to 2048, 8, and 1e-4, respectively, and finally the model was trained for 5 epochs. To conclude, it is possible to develop a good extractive summarization system for Turkish news using the proposed approach.

As managerial implications, summarization systems are quite useful in the internet era since huge amounts of textual information are broadcasted every time via social media, online news websites or blogs. By developing automatic summarization systems, users or readers can access required, significant, and actionable information and these systems reduce the reading time and the time to access the important and relevant information. These systems can be beneficial for various fields. Extractive summarization systems can be used in generating summaries of long business reports, legal documents, academic papers or news, automatically. The model proposed in this thesis can be applied easily to the real-world applications.

CHAPTER 6

FURTHER RESEARCH

In this thesis, the promising results have been achieved since the proposed model reached higher ROUGE scores than the LEAD-5 baseline. This is important since the LEAD-5 score is very high. The reason why the LEAD-5 score is so high is that the collected Turkish news contains the important contents in the very first sentences. In other words, the human-written (abstractive) summaries are written by considering mostly the first sentences of the main articles as seen in Figure 9. However, there are further research opportunities to increase the performance results of extractive summarization systems for Turkish news.

Firstly, the gathered data set size is comparably small, which has 2076 news and 1476 of them are used for training the model. In the news website the data collected, there were no more news and another website could not be found, which has human-written summaries for Turkish news. For comparison, CNN/Daily Mail dataset has around 300K news (See et al., 2017). Although collecting much more news articles with the relevant human written summaries may lead to longer training times and also, may require higher memory usage and processing power, it is possible to train more robust and generalizable summarization models for Turkish news.

Secondly, extractive summarization systems are directly copying the significant and salient sentences occurred in the main articles to generate final summaries. However, most of the summarization datasets contain the abstractive summaries. To train extractive summarization models, the sentences in the main articles should be labelled based on these abstractive summaries. In other words, the extracted sentences

from main articles should contain the most or all of the information given by abstractive summaries. In the literature, some heuristic and rule-based approaches are commonly used for this label extraction process. The most common one is the greedy approach proposed by Nallapati et al. (2016a). This approach selects the sentences so that the selected sentence set has the highest possible ROUGE-2 F scores with respect to the abstractive summaries. In this thesis, this approach is utilized to determine sentence labels. However, this approach is rule-based and does not guarantee the most suitable labels (Narayan et al., 2018). In the future research, this process can be done by human annotators similar to the work of Cheng and Lapata (2016). The main articles are shown to the annotators and they can select the most informative sentences in the main articles. The selected sentences are labelled as 1 and the others are labelled as 0. Then, these labels can be used to train the summarization models properly, despite the fact that this annotation process may be more costly and time consuming for researchers.

Another future research can be studied for Turkish news extractive summarization might investigate other pre-trained language models, which are trained with Turkish corpus. During the time this study was conducted, there were only BERT (Devlin et al., 2018) and ELECTRA (Clark et al., 2020) for Turkish. However, in the future, the other popular language models such as RoBERTa can be trained for Turkish and these language models can perform better.

Finally, in this thesis, the sentences are selected individually based on their scores and high scoring sentences generated a final summary. However, this may lead to overlooking better candidates since the best possible summaries may not include only high scoring sentences as explained by Zhong et al. (2020). They applied Siamese networks to measure and learn the similarities between candidate summaries and main

articles, as mentioned in Section 2.3. However, training Siamese networks require huge memory (RAM) and processing powers (GPU). For example, the MatchSUM model was trained with 8 16GB GPUs and the training took 30 hours with these hardware (Zhong et al., 2020). In this study, these resources are not accessible and affordable. But, the researchers, who have access to required hardware, can train the Siamese networks to achieve better performance scores for Turkish news.

APPENDIX A

PERFORMANCE RESULTS OF ALL MODELS

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
1	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	2048	8	4	5.00E-05	00:01:31	37.46	26.2	37.13
2	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	2048	8	4	1.00E-04	00:01:31	37.47	26.21	37.11
3	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	2048	8	4	2.00E-03	00:01:30	37.48	26.13	37.09
4	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	3072	12	4	5.00E-05	00:01:28	37.6	26.21	37.24
5	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	3072	12	4	1.00E-04	00:01:32	37.71	26.41	37.36
6	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	3072	12	4	2.00E-03	00:01:31	37.44	25.87	36.96
7	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	4	5.00E-05	00:01:36	37.63	26.42	37.28
8	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	4	1.00E-04	00:01:35	37.64	26.41	37.32
9	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	4	2.00E-03	00:01:34	37.46	26.12	37.08
10	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	4	5.00E-05	00:01:33	37.58	26.33	37.22
11	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	4	1.00E-04	00:01:36	37.72	26.49	37.36

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
12	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	4	2.00E-03	00:01:37	37.53	26.26	37.14
13	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	5.00E-05	00:01:40	37.62	26.44	37.27
14	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	1.00E-04	00:01:40	37.66	26.45	37.3
15	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	2.00E-03	00:01:37	37.64	26.16	37.18
16	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	3072	12	4	5.00E-05	00:01:39	37.51	26.25	37.16
17	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	3072	12	4	1.00E-04	00:01:38	37.78	26.5	37.43
18	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	3072	12	4	2.00E-03	00:01:40	37.57	26.24	37.19
19	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	2048	8	5	5.00E-05	00:01:53	37.59	26.23	37.18
20	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	2048	8	5	1.00E-04	00:01:51	37.69	26.34	37.31
21	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	2048	8	5	2.00E-03	00:01:48	27.4	15.94	26.75
22	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	3072	12	5	5.00E-05	00:01:51	37.56	26.15	37.21
23	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	3072	12	5	1.00E-04	00:01:53	37.74	26.35	37.4
24	BERTurk-Base (32K)	CLS Token of the Last Layer	1-Layer Transformer	3072	12	5	2.00E-03	00:01:53	37.48	26.15	37.11
25	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	5.00E-05	00:03:12	37.69	26.43	37.33

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
26	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	1.00E-04	00:02:01	37.69	26.46	37.38
27	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	2.00E-03	00:02:00	37.51	26.08	37.09
28	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	5.00E-05	00:01:58	37.8	26.53	37.45
29	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	1.00E-04	00:01:58	37.71	26.49	37.37
30	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	2.00E-03	00:01:56	37.42	26.12	37.05
31	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	5.00E-05	00:02:06	37.68	26.43	37.31
32	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	1.00E-04	00:02:05	37.8	26.47	37.41
33	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	2.00E-03	00:02:01	37.42	25.94	37.03
34	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	3072	12	5	5.00E-05	00:02:08	37.6	26.32	37.25
35	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	3072	12	5	1.00E-04	00:02:03	37.66	26.33	37.29
36	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	3072	12	5	2.00E-03	00:02:00	37.57	26.21	37.18
37	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	4	5.00E-05	00:01:50	37.41	26.15	37.05
38	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	4	1.00E-04	00:01:48	37.4	26.16	37.04
39	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	4	2.00E-03	00:01:49	27.4	15.94	26.75

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
40	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	4	5.00E-05	00:01:52	37.49	26.27	37.11
41	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	4	1.00E-04	00:01:48	37.48	26.3	37.09
42	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	4	2.00E-03	00:01:51	37.46	26.1	37.06
43	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	4	5.00E-05	00:01:51	37.39	26.22	37.03
44	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	4	1.00E-04	00:01:52	37.4	26.2	37.04
45	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	4	2.00E-03	00:01:53	37.4	26.02	37.01
46	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	4	5.00E-05	00:01:54	37.43	26.21	37.08
47	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	4	1.00E-04	00:01:52	37.45	26.26	37.1
48	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	4	2.00E-03	00:01:54	37.49	26.39	37.12
49	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	4	5.00E-05	00:01:55	37.38	26.22	37.02
50	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	4	1.00E-04	00:01:53	37.42	26.25	37.07
51	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	4	2.00E-03	00:01:53	37.42	26.19	37.05
52	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	4	5.00E-05	00:01:58	37.47	26.31	37.1
53	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	4	1.00E-04	00:01:57	37.41	26.25	37.06

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
54	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	4	2.00E-03	00:01:57	37.5	26.36	37.15
55	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	5	5.00E-05	00:02:16	37.52	26.24	37.17
56	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	5	1.00E-04	00:02:16	37.42	26.21	37.06
57	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	5	2.00E-03	00:02:16	27.4	15.94	26.75
58	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	5	5.00E-05	00:02:15	37.47	26.21	37.1
59	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	5	1.00E-04	00:02:14	37.41	26.21	37.06
60	BERTurk-Base (32K)	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	5	2.00E-03	00:02:14	37.49	26.36	37.12
61	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	5	5.00E-05	00:02:20	37.4	26.16	37.04
62	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	5	1.00E-04	00:02:19	37.61	26.31	37.26
63	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	5	2.00E-03	00:02:20	37.41	26.22	37.03
64	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	5	5.00E-05	00:02:20	37.61	26.33	37.26
65	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	5	1.00E-04	00:02:20	37.59	26.34	37.22
66	BERTurk-Base (32K)	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	5	2.00E-03	00:02:20	37.39	26.23	37
67	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	5	5.00E-05	00:02:25	37.41	26.24	37.07

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
68	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	5	1.00E-04	00:02:25	37.51	26.28	37.19
69	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	5	2.00E-03	00:02:25	37.42	25.99	37.06
70	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	5	5.00E-05	00:02:25	37.43	26.23	37.08
71	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	5	1.00E-04	00:02:25	37.52	26.23	37.18
72	BERTurk-Base (32K)	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	5	2.00E-03	00:02:25	37.5	26.33	37.14
73	BERTurk-Base (32K)	CLS Token of the Last Layer	Simple Linear Layer	-	-	10	5.00E-05	00:06:09	31.45	19.82	30.85
74	BERTurk-Base (32K)	CLS Token of the Last Layer	Simple Linear Layer	-	-	10	1.00E-04	00:06:10	32.58	20.85	32.09
75	BERTurk-Base (32K)	CLS Token of the Last Layer	Simple Linear Layer	-	-	10	2.00E-03	00:06:10	36.1	24.46	35.81
76	BERTurk-Base (32K)	Mean pooling of the Last Layer	Simple Linear Layer	-	-	10	5.00E-05	00:04:25	32.62	20.24	32
77	BERTurk-Base (32K)	Mean pooling of the Last Layer	Simple Linear Layer	-	-	10	1.00E-04	00:04:24	33.8	21.37	33.29
78	BERTurk-Base (32K)	Mean pooling of the Last Layer	Simple Linear Layer	-	-	10	2.00E-03	00:04:24	36.92	24.85	36.56
79	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	2048	8	4	5.00E-05	00:02:30	37.49	26.4	37.12
80	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	2048	8	4	1.00E-04	00:02:30	37.49	26.4	37.12
81	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	2048	8	4	2.00E-03	00:02:32	37.45	26.2	37.1

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
82	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	3072	12	4	5.00E-05	00:02:31	37.49	26.4	37.12
83	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	3072	12	4	1.00E-04	00:02:32	37.49	26.4	37.12
84	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	3072	12	4	2.00E-03	00:02:32	37.49	26.4	37.12
85	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	2048	8	4	5.00E-05	00:02:36	37.49	26.4	37.12
86	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	2048	8	4	1.00E-04	00:02:36	37.49	26.4	37.12
87	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	2048	8	4	2.00E-03	00:02:36	37.25	25.42	36.82
88	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	3072	12	4	5.00E-05	00:02:36	37.49	26.4	37.12
89	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	3072	12	4	1.00E-04	00:02:36	37.49	26.4	37.12
90	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	3072	12	4	2.00E-03	00:02:35	37.49	26.26	37.11
91	ELEC**TR**A	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	5.00E-05	00:02:39	37.49	26.4	37.12
92	ELEC**TR**A	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	1.00E-04	00:02:39	37.49	26.4	37.12
93	ELEC**TR**A	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	2.00E-03	00:02:38	27.4	15.94	26.75
94	ELEC**TR**A	CLS Token of the Last Layer	3-Layer Transformers	3072	12	4	5.00E-05	00:02:39	37.49	26.4	37.12
95	ELEC**TR**A	CLS Token of the Last Layer	3-Layer Transformers	3072	12	4	1.00E-04	00:02:38	37.49	26.4	37.12

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
96	ELEC**TR**A	CLS Token of the Last Layer	3-Layer Transformers	3072	12	4	2.00E-03	00:02:38	27.4	15.94	26.75
97	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	2048	8	5	5.00E-05	00:01:53	37.49	26.4	37.12
98	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	2048	8	5	1.00E-04	00:01:53	37.49	26.4	37.12
99	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	2048	8	5	2.00E-03	00:01:55	37.41	25.73	37.02
100	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	3072	12	5	5.00E-05	00:01:56	37.49	26.4	37.12
101	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	3072	12	5	1.00E-04	00:01:55	37.49	26.4	37.12
102	ELEC**TR**A	CLS Token of the Last Layer	1-Layer Transformer	3072	12	5	2.00E-03	00:01:55	37.49	26.4	37.12
103	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	5.00E-05	00:02:01	37.49	26.4	37.12
104	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	1.00E-04	00:02:01	37.49	26.4	37.12
105	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	2.00E-03	00:02:01	37.49	26.25	37.13
106	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	5.00E-05	00:02:01	37.49	26.4	37.12
107	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	1.00E-04	00:02:01	37.49	26.4	37.12
108	ELEC**TR**A	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	2.00E-03	00:02:01	37.42	26.32	37.04
109	ELEC**TR**A	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	5.00E-05	00:02:07	37.49	26.4	37.12

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
110	ELECTRA	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	1.00E-04	00:02:07	37.49	26.4	37.12
111	ELECTRA	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	2.00E-03	00:02:06	27.4	15.94	26.75
112	ELECTRA	CLS Token of the Last Layer	3-Layer Transformers	3072	12	5	5.00E-05	00:02:07	37.49	26.4	37.12
113	ELECTRA	CLS Token of the Last Layer	3-Layer Transformers	3072	12	5	1.00E-04	00:02:07	37.49	26.4	37.12
114	ELECTRA	CLS Token of the Last Layer	3-Layer Transformers	3072	12	5	2.00E-03	00:02:06	27.4	15.94	26.75
115	ELECTRA	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	4	5.00E-05	00:01:52	37.49	26.39	37.12
116	ELECTRA	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	4	1.00E-04	00:01:51	37.49	26.39	37.12
117	ELECTRA	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	4	2.00E-03	00:01:52	37.51	25.92	37.13
118	ELECTRA	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	4	5.00E-05	00:02:51	37.49	26.35	37.12
119	ELECTRA	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	4	1.00E-04	00:02:50	37.49	26.4	37.12
120	ELECTRA	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	4	2.00E-03	00:02:50	27.4	15.94	26.75
121	ELECTRA	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	4	5.00E-05	00:02:56	37.49	26.4	37.12
122	ELECTRA	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	4	1.00E-04	00:02:57	37.49	26.4	37.12
123	ELECTRA	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	4	2.00E-03	00:02:55	27.4	15.94	26.75

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
124	ELEC**TR**A	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	4	5.00E-05	00:02:55	37.49	26.39	37.12
125	ELEC**TR**A	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	4	1.00E-04	00:02:55	37.49	26.39	37.12
126	ELEC**TR**A	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	4	2.00E-03	00:02:55	27.4	15.94	26.75
127	ELEC**TR**A	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	4	5.00E-05	00:02:59	37.49	26.4	37.12
128	ELEC**TR**A	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	4	1.00E-04	00:02:59	37.49	26.39	37.12
129	ELEC**TR**A	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	4	2.00E-03	00:03:01	27.4	15.94	26.75
130	ELEC**TR**A	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	4	5.00E-05	00:03:01	37.49	26.39	37.12
131	ELEC**TR**A	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	4	1.00E-04	00:03:00	37.49	26.39	37.12
132	ELEC**TR**A	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	4	2.00E-03	00:03:00	27.4	15.94	26.75
133	ELEC**TR**A	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	5	5.00E-05	00:03:34	37.49	26.39	37.12
134	ELEC**TR**A	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	5	1.00E-04	00:03:34	37.49	26.4	37.12
135	ELEC**TR**A	Mean pooling of the Last Layer	1-Layer Transformer	2048	8	5	2.00E-03	00:03:34	37.49	26.36	37.12
136	ELEC**TR**A	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	5	5.00E-05	00:03:33	37.49	26.39	37.12
137	ELEC**TR**A	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	5	1.00E-04	00:03:33	37.49	26.39	37.12

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
138	ELECTRA	Mean pooling of the Last Layer	1-Layer Transformer	3072	12	5	2.00E-03	00:03:34	37.49	26.38	37.12
139	ELECTRA	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	5	5.00E-05	00:03:39	37.49	26.39	37.12
140	ELECTRA	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	5	1.00E-04	00:03:38	37.49	26.39	37.12
141	ELECTRA	Mean pooling of the Last Layer	2-Layer Transformers	2048	8	5	2.00E-03	00:03:39	37.49	26.35	37.12
142	ELECTRA	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	5	5.00E-05	00:03:39	37.49	26.38	37.12
143	ELECTRA	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	5	1.00E-04	00:03:38	37.49	26.38	37.12
144	ELECTRA	Mean pooling of the Last Layer	2-Layer Transformers	3072	12	5	2.00E-03	00:03:37	27.4	15.94	26.75
145	ELECTRA	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	5	5.00E-05	00:03:48	37.49	26.39	37.12
146	ELECTRA	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	5	1.00E-04	00:03:53	37.49	26.39	37.12
147	ELECTRA	Mean pooling of the Last Layer	3-Layer Transformers	2048	8	5	2.00E-03	00:03:49	27.4	15.94	26.75
148	ELECTRA	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	5	5.00E-05	00:03:48	37.49	26.39	37.12
149	ELECTRA	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	5	1.00E-04	00:03:51	37.49	26.38	37.12
150	ELECTRA	Mean pooling of the Last Layer	3-Layer Transformers	3072	12	5	2.00E-03	00:03:47	27.4	15.94	26.75
151	ELECTRA	CLS Token of the Last Layer	Simple Linear Layer	-	-	10	5.00E-05	00:06:04	27.35	15.89	26.71

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
152	ELEC**TR**A	CLS Token of the Last Layer	Simple Linear Layer	-	-	10	1.00E-04	00:06:03	27.35	15.89	26.71
153	ELEC**TR**A	CLS Token of the Last Layer	Simple Linear Layer	-	-	10	2.00E-03	00:06:03	27.21	15.78	26.54
154	ELEC**TR**A	Mean pooling of the Last Layer	Simple Linear Layer	-	-	10	5.00E-05	00:04:12	28.72	16.65	28.15
155	ELEC**TR**A	Mean pooling of the Last Layer	Simple Linear Layer	-	-	10	1.00E-04	00:04:10	29.62	17.76	29.12
156	ELEC**TR**A	Mean pooling of the Last Layer	Simple Linear Layer	-	-	10	2.00E-03	00:04:09	32.38	20.62	32.03
157	BERTurk-Base (32K)	CLS Token + Mean pooling	2-Layer Transformers	3072	12	5	5.00E-05	00:02:23	37.73	26.25	37.4
158	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer Transformers	3072	12	4	1.00E-04	00:02:01	37.91	26.46	37.5
159	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer Transformers	2048	8	5	1.00E-04	00:02:28	38.02	26.49	37.62
160	BERTurk-Base (32K)	CLS Token + Mean pooling	2-Layer Transformers	2048	8	5	1.00E-04	00:02:24	38.38	26.8	38.04
161	BERTurk-Base (32K)	CLS Token + Mean pooling	2-Layer Transformers	3072	12	5	1.00E-04	00:02:18	37.76	26.28	37.45
162	BERTurk-Base (32K)	CLS Token + Mean pooling	2-Layer Transformers	3072	12	4	1.00E-04	00:01:50	37.76	26.33	37.43
163	BERTurk-Base (32K)	CLS Token + Mean pooling	2-Layer Transformers	2048	8	5	5.00E-05	00:03:32	37.98	26.44	37.59
164	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer Transformers	2048	8	4	1.00E-04	00:01:56	38.03	26.55	37.66
165	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer Transformers	2048	8	4	5.00E-05	00:02:55	37.88	26.42	37.49

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
166	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer Transformers	2048	8	5	5.00E-05	00:03:37	38.03	26.5	37.6
167	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	3072	12	5	5.00E-05	00:02:22	37.42	26.05	37.08
168	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	3072	12	4	1.00E-04	00:01:59	37.72	26.45	37.32
169	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	2048	8	5	1.00E-04	00:02:29	37.44	26.03	37.12
170	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	2048	8	5	1.00E-04	00:03:55	37.52	26.17	37.2
171	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	3072	12	5	1.00E-04	00:03:38	37.36	25.9	37.03
172	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	3072	12	4	1.00E-04	00:01:54	37.73	26.48	37.35
173	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	2048	8	5	5.00E-05	00:02:19	37.48	26.23	37.16
174	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	2048	8	4	1.00E-04	00:03:00	37.7	26.54	37.36
175	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	2048	8	4	5.00E-05	00:03:00	37.75	26.56	37.38
176	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	2048	8	5	5.00E-05	00:02:28	37.52	26.17	37.12
177	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	6144	24	5	5.00E-05	00:02:23	37.52	26.28	37.15
178	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	6144	24	4	1.00E-04	00:02:02	27.4	15.94	26.75
179	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	4096	16	5	1.00E-04	00:02:30	37.49	26.16	37.17

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
180	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	4096	16	5	1.00E-04	00:02:21	37.42	25.99	37.05
181	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	6144	24	5	1.00E-04	00:02:23	37.51	26.23	37.14
182	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	6144	24	4	1.00E-04	00:01:56	37.69	26.45	37.34
183	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	4096	16	5	5.00E-05	00:02:21	37.65	26.37	37.31
184	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	4096	16	4	1.00E-04	00:01:55	37.83	26.57	37.47
185	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	4096	16	4	5.00E-05	00:01:55	37.67	26.39	37.33
186	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	4096	16	5	5.00E-05	00:02:23	37.41	26.1	37.09
187	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	2-Layer Transformers	3072	12	5	5.00E-05	00:02:28	37.63	26.29	37.29
188	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	3072	12	4	1.00E-04	00:02:04	37.49	26.38	37.12
189	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	2048	8	5	1.00E-04	00:04:17	37.5	26.4	37.12
190	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	2-Layer Transformers	2048	8	5	1.00E-04	00:03:45	27.4	15.94	26.75
191	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	2-Layer Transformers	3072	12	5	1.00E-04	00:02:27	37.53	26.41	37.18
192	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	2-Layer Transformers	3072	12	4	1.00E-04	00:01:55	37.49	26.31	37.13
193	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	2-Layer Transformers	2048	8	5	5.00E-05	00:03:48	37.36	26.07	37.04

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
194	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	2048	8	4	1.00E-04	00:02:02	37.44	26.28	37.07
195	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	2048	8	4	5.00E-05	00:02:02	37.56	26.41	37.2
196	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	2048	8	5	5.00E-05	00:02:33	37.74	26.45	37.39
197	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	2-Layer Transformers	9216	36	5	5.00E-05	00:02:39	37.52	26.22	37.21
198	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	9216	36	4	1.00E-04	00:02:21	37.49	26.41	37.12
199	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	6144	24	5	1.00E-04	00:02:47	37.67	26.41	37.32
200	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	2-Layer Transformers	6144	24	5	1.00E-04	00:02:32	37.49	26.2	37.1
201	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	2-Layer Transformers	9216	36	5	1.00E-04	00:02:41	37.49	26.29	37.16
202	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	2-Layer Transformers	9216	36	4	1.00E-04	00:02:06	37.48	26.34	37.11
203	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	2-Layer Transformers	6144	24	5	5.00E-05	00:02:31	37.67	26.29	37.33
204	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	6144	24	4	1.00E-04	00:02:12	37.48	26.25	37.09
205	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	6144	24	4	5.00E-05	00:02:12	37.71	26.5	37.34
206	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	6144	24	5	5.00E-05	00:02:43	37.85	26.59	37.53
207	BERTurk-Base (32K)	Sum Last Four Layers' CLS Token	2-Layer Transformers	3072	12	5	5.00E-05	00:03:34	37.76	26.39	37.53

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
208	BERTurk-Base (32K)	Sum Last Four Layers' CLS Token	3-Layer Transformers	3072	12	4	1.00E-04	00:02:55	37.64	26.16	37.38
209	BERTurk-Base (32K)	Sum Last Four Layers' CLS Token	3-Layer Transformers	2048	8	5	1.00E-04	00:03:40	37.69	26.07	37.32
210	BERTurk-Base (32K)	Sum Last Four Layers' CLS Token	2-Layer Transformers	2048	8	5	1.00E-04	00:03:34	37.64	26.22	37.36
211	BERTurk-Base (32K)	Sum Last Four Layers' CLS Token	2-Layer Transformers	3072	12	5	1.00E-04	00:03:34	37.68	26.17	37.35
212	BERTurk-Base (32K)	Sum Last Four Layers' CLS Token	2-Layer Transformers	3072	12	4	1.00E-04	00:02:51	37.58	26.13	37.23
213	BERTurk-Base (32K)	Sum Last Four Layers' CLS Token	2-Layer Transformers	2048	8	5	5.00E-05	00:03:34	37.92	26.37	37.58
214	BERTurk-Base (32K)	Sum Last Four Layers' CLS Token	3-Layer Transformers	2048	8	4	1.00E-04	00:02:55	37.59	26.09	37.21
215	BERTurk-Base (32K)	Sum Last Four Layers' CLS Token	3-Layer Transformers	2048	8	4	5.00E-05	00:02:55	37.62	26.11	37.23
216	BERTurk-Base (32K)	Sum Last Four Layers' CLS Token	3-Layer Transformers	2048	8	5	5.00E-05	00:03:40	37.7	26.18	37.35
217	BERTurk-Base (128K)	CLS Token + Mean pooling	2-Layer Transformers	2048	8	5	1.00E-04	00:02:14	37.83	26.42	37.51
218	BERTurk-Base (128K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	6144	24	5	5.00E-05	00:02:45	37.62	26.39	37.27
219	BERTurk-Base (128K)	(CLS Token, Mean pooling)	3-Layer Transformers	4096	16	4	1.00E-04	00:01:58	27.33	15.88	26.66
220	BERTurk-Base (128K)	(CLS Token, Mean pooling)	3-Layer Transformers	2048	8	4	5.00E-05	00:01:57	37.49	26.4	37.12
221	BERTurk-Base (128K)	CLS Token + Mean pooling	3-Layer Transformers	2048	8	4	1.00E-04	00:01:51	37.61	26.22	37.27

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
222	BERTurk-Base (128K)	(CLS Token, Mean pooling|)	3-Layer Transformers	2048	8	4	1.00E-04	00:01:54	27.33	15.88	26.66
223	BERTurk-Base (128K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	5.00E-05	00:01:58	37.49	26.4	37.12
224	BERTurk-Base (128K)	CLS Token of the Last Layer	3-Layer Transformers	3072	12	4	1.00E-04	00:01:38	37.49	26.4	37.12
225	BERTurk-Base (128K)	CLS Token + Mean pooling	3-Layer Transformers	2048	8	5	5.00E-05	00:02:18	37.6	26.25	37.27
226	BERTurk-Base (128K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	6144	24	4	5.00E-05	00:02:12	37.49	26.34	37.13
227	BERTurk-Base (128K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	4	1.00E-04	00:01:31	37.49	26.4	37.12
228	BERTurk-Base (128K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	1.00E-04	00:01:55	37.49	26.4	37.12
229	BERTurk-Base (128K)	CLS Token + Mean pooling	3-Layer Transformers	2048	8	5	1.00E-04	00:02:19	37.78	26.37	37.46
230	BERTurk-Base (128K)	(CLS Token, Mean pooling)	2-Layer Transformers	3072	12	4	1.00E-04	00:01:51	37.49	26.35	37.12
231	BERTurk-Base (128K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	1.00E-04	00:01:59	37.49	26.4	37.12
232	BERTurk-Base (128K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	1.00E-04	00:01:54	37.49	26.4	37.12
233	BERTurk-Base (128K)	CLS Token + Mean pooling	3-Layer Transformers	3072	12	4	1.00E-04	00:01:51	37.6	26.22	37.26
234	BERTurk-Base (128K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	1.00E-04	00:01:35	37.49	26.4	37.12
235	BERTurk-Base (128K)	(CLS Token, Mean pooling)	3-Layer Transformers	3072	12	4	1.00E-04	00:01:55	37.49	26.39	37.12

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
236	BERTurk-Base (128K)	(CLS Token, Mean pooling)	2-Layer Transformers	6144	24	4	1.00E-04	00:01:53	37.49	26.35	37.12
237	BERTurk-Base (128K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	2048	8	5	5.00E-05	00:02:33	37.51	26.28	37.16
238	BERTurk-Base (128K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	5.00E-05	00:01:35	37.49	26.4	37.12
239	BERTurk-Base (128K)	CLS Token + Mean pooling	2-Layer Transformers	2048	8	5	5.00E-05	00:02:14	37.71	26.29	37.39
240	BERTurk-Base (128K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	5.00E-05	00:02:01	37.49	26.4	37.12
241	BERTurk-Base (128K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	5.00E-05	00:02:07	37.49	26.4	37.12

APPENDIX B

TOP-10 PERFORMING MODELS

AMONG THE INITIAL 158 MODELS

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
28	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	5.00E-05	0:01:58	37.8	26.53	37.45
17	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	3072	12	4	1.00E-04	0:01:38	37.78	26.5	37.43
11	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	4	1.00E-04	0:01:36	37.72	26.49	37.36
29	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	1.00E-04	0:01:58	37.71	26.49	37.37
32	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	1.00E-04	0:02:05	37.8	26.47	37.41
26	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	1.00E-04	0:02:01	37.69	26.46	37.38
14	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	1.00E-04	0:01:40	37.66	26.45	37.3
13	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	5.00E-05	0:01:40	37.62	26.44	37.27
25	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	5.00E-05	0:03:12	37.69	26.43	37.33
31	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	5.00E-05	0:02:06	37.68	26.43	37.31

APPENDIX C

TOP-25 BEST PERFORMING BERTURK BASE (32K) MODELS' HYPERPARAMETER SETTINGS USED IN BERTURK BASE (128) MODEL EXPERIMENTS

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
160	BERTurk-Base (32K)	CLS Token + Mean pooling	2-Layer Transformers	2048	8	5	1.00E-04	0:02:24	38.38	26.8	38.04
206	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	6144	24	5	5.00E-05	0:02:43	37.85	26.59	37.53
184	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	4096	16	4	1.00E-04	0:01:55	37.83	26.57	37.47
175	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	2048	8	4	5.00E-05	0:03:00	37.75	26.56	37.38
164	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer Transformers	2048	8	4	1.00E-04	0:01:56	38.03	26.55	37.66
174	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	2048	8	4	1.00E-04	0:03:00	37.7	26.54	37.36
28	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	5.00E-05	0:01:58	37.8	26.53	37.45
17	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	3072	12	4	1.00E-04	0:01:38	37.78	26.5	37.43
166	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer Transformers	2048	8	5	5.00E-05	0:03:37	38.03	26.5	37.6
205	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	6144	24	4	5.00E-05	0:02:12	37.71	26.5	37.34
11	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	4	1.00E-04	0:01:36	37.72	26.49	37.36

#	Language Model	Sentence Representation	Extra Layer	FFN Hidden Size	# of Attention Heads	# of Epoch	Learning Rate	Training Time	ROUGE-1 F score	ROUGE-2 F score	ROUGE-L F score
29	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	3072	12	5	1.00E-04	0:01:58	37.71	26.49	37.37
159	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer Transformers	2048	8	5	1.00E-04	0:02:28	38.02	26.49	37.62
172	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	3072	12	4	1.00E-04	0:01:54	37.73	26.48	37.35
32	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	1.00E-04	0:02:05	37.8	26.47	37.41
26	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	1.00E-04	0:02:01	37.69	26.46	37.38
158	BERTurk-Base (32K)	CLS Token + Mean pooling	3-Layer Transformers	3072	12	4	1.00E-04	0:02:01	37.91	26.46	37.5
14	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	1.00E-04	0:01:40	37.66	26.45	37.3
168	BERTurk-Base (32K)	(CLS Token, Mean pooling)	3-Layer Transformers	3072	12	4	1.00E-04	0:01:59	37.72	26.45	37.32
182	BERTurk-Base (32K)	(CLS Token, Mean pooling)	2-Layer Transformers	6144	24	4	1.00E-04	0:01:56	37.69	26.45	37.34
196	BERTurk-Base (32K)	(CLS Token, Mean pooling, |CLS Token-Mean pooling|)	3-Layer Transformers	2048	8	5	5.00E-05	0:02:33	37.74	26.45	37.39
13	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	4	5.00E-05	0:01:40	37.62	26.44	37.27
163	BERTurk-Base (32K)	CLS Token + Mean pooling	2-Layer Transformers	2048	8	5	5.00E-05	0:03:32	37.98	26.44	37.59
25	BERTurk-Base (32K)	CLS Token of the Last Layer	2-Layer Transformers	2048	8	5	5.00E-05	0:03:12	37.69	26.43	37.33
31	BERTurk-Base (32K)	CLS Token of the Last Layer	3-Layer Transformers	2048	8	5	5.00E-05	0:02:06	37.68	26.43	37.31

REFERENCES

Bae, S., Kim, T., Kim, J., & Lee, S. (2019). Summary level training of sentence rewriting for abstractive summarization. *Proceedings of the 2nd Workshop on New Frontiers in Summarization*. doi:10.18653/v1/d19-5402

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: Analyzing text with the natural language toolkit*. Newton, MA: O'Reilly Media.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with Subword information. *Transactions of the Association for Computational Linguistics, 5*, 135-146. doi:10.1162/tacl_a_00051

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., & Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. Retrieved from https://arxiv.org/abs/1312.3005

Cheng, J., & Lapata, M. (2016). Neural summarization by extracting sentences and words. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. doi:10.18653/v1/p16-1046

Cho, K., Van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. doi:10.3115/v1/d14-1179

Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. Retrieved from https://arxiv.org/abs/2003.10555

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. Retrieved from https://arxiv.org/abs/1810.04805

Dong, Y., Shen, Y., Crawford, E., Van Hoof, H., & Cheung, J. C. (2018). BanditSum: Extractive summarization as a contextual bandit. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. doi:10.18653/v1/d18-1409

Goldberg, Y. (2017). *Neural network methods in natural language processing (synthesis lectures on human language technologies).* Williston, VT: Morgan & Claypool Publishers.

Guo, J., Che, W., Wang, H., & Liu, T. (2014). Revisiting embedding features for simple semi-supervised learning. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. doi:10.3115/v1/d14-1012

Guo, W., Wu, B., Wang, B., & Yang, Y. (2020). Two-stage encoding extractive summarization. *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*. doi:10.1109/dsc50466.2020.00060

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735-1780. Retrieved from https://doi.org/10.1162/neco.1997.9.8.1735

Kingma, D., & Ba, J. L. (2014). Adam: A method for stochastic optimization. Retrieved from https://arxiv.org/abs/1412.6980

Lin, C. Y. (2004). ROUGE: A package for automatic evaluation of summaries. Retrieved from https://www.aclweb.org/anthology/W04-1013.pdf

Liu, Y., & Lapata, M. (2019). Text summarization with pretrained encoders. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. doi:10.18653/v1/d19-1387

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., … Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. Retrieved from https://arxiv.org/abs/1907.11692

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. Retrieved from https://arxiv.org/abs/1301.3781

Nallapati, R., Zhai, F., & Zhou, B. (2016a). SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. Retrieved from https://arxiv.org/abs/1611.04230

Nallapati, R., Zhou, B., Dos Santos, C., Gulcehre, C., & Xiang, B. (2016b). Abstractive text summarization using sequence-to-sequence RNNs and beyond. *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. doi:10.18653/v1/k16-1028

Narayan, S., Cohen, S. B., & Lapata, M. (2018). Ranking sentences for extractive summarization with reinforcement learning. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. doi:10.18653/v1/n18-1158

Paulus, R., Xiong, C., & Socher, R. (2017). A deep reinforced model for abstractive summarization. Retrieved from https://arxiv.org/abs/1705.04304

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. doi:10.3115/v1/d14-1162

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. Retrieved from https://arxiv.org/abs/1802.05365

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. doi:10.18653/v1/d19-1410

Ruder, S. (2020). Tracking progress in natural language processing. Retrieved October 2020, from https://github.com/sebastianruder/NLP-progress

Schweter, S. (2020, April). BERTurk - BERT models for Turkish. Retrieved October 2020, from https://github.com/stefan-it/turkish-bert

Scialom, T., Dray, P. A., Lamprier, S., Piwowarski, B., & Staiano, J. (2020). MLSUM: The multilingual summarization corpus. Retrieved from https://arxiv.org/abs/2004.14900

See, A., Liu, E. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. Retrieved from https://arxiv.org/abs/1704.04368

Sonmez, S. (2016, February 24). "Facebook'un geleceği yapay zekada". Retrieved from https://www.dunyahalleri.com/facebookun-gelecegi-yapay-zekada/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 5998-6008. Retrieved from https://arxiv.org/abs/1706.03762

Zhang, X., Wei, F., & Zhou, M. (2019). HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. doi:10.18653/v1/p19-1499

Zhong, M., Liu, P., Chen, Y., Wang, D., Qiu, X., & Huang, X. (2020). Extractive Summarization as text matching. Retrieved from https://arxiv.org/abs/2004.08795

Zhong, M., Liu, P., Wang, D., Qiu, X., & Huang, X. (2019). Searching for effective neural extractive summarization: What works and what's next. Retrieved from https://arxiv.org/abs/1907.03491

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *Proceedings of the IEEE international conference on computer vision*, 19-27. Retrieved from https://arxiv.org/abs/1506.06724