SIMILAR QA PAIRS DETECTION SYSTEM TO IMPROVE CHAT QUALITY IN TURKISH CONVERSATION GROUPS

TÜRKÇE KONUŞMA GRUPLARINDA SOHBET KALİTESİNİ ARTIRMAK İÇİN BENZER SORU-CEVAP ÇİFTİ TESPİT SİSTEMİ

İZZET KILIÇ

ASSOC. PROF. DERYA KARAGÖZ
Supervisor

Submitted to
Graduate School of Science and Engineering of Hacettepe University
as a Partial Fulfillment to the Requirements
for the Award of the Degree of Master of Science
in Statistics

May 2022

ABSTRACT

SIMILAR QA PAIRS DETECTION SYSTEM TO IMPROVE CHAT QUALITY IN TURKISH CONVERSATION GROUPS

İzzet Kılıç
Master of Science, Statistics
Supervisor: Assoc. Prof. Derya Karagöz
May 2022, 92 pages

Nowadays, social media applications have become one of our essential communication tools. These applications can be used to communicate individually or in groups. Retrieving information by processing the text data produced in user groups increases the quality of the chat. Consequently, creating a system that will prevent the same question from being asked multiple times in groups with thousands of users will help increase efficiency in the chat. With the help of the application developed on the Slack platform in this study, users can search for similar questions in old messages and retrieve their answers. In cases where there is no similar question in the conversation, a mechanism has been developed to direct the question that is asked by a user to a user who has knowledge of the subject. The BERTurk model is fine-tuned for sentence classification and question answering tasks, which are natural language processing techniques. Since a large amount of training data is needed, datasets created in other languages are translated to Turkish and open-source datasets are used.

Keywords: Natural Language Processing, Deep Learning, Question Answering, Question Detection

ÖZET

TÜRKÇE KONUŞMA GRUPLARINDA SOHBET KALİTESİNİ ARTIRMAK İÇİN BENZER SORU-CEVAP ÇİFTİ TESPİT SİSTEMİ

İzzet Kılıç
Yüksek Lisans, İstatistik Bölümü
Tez Danışman: Doç. Dr. Derya Karagöz
Mayıs 2022, 92 sayfa

Günümüzde sosyal medya uygulamaları temel iletişim araçlarımızdan biri haline gelmiştir. Bu uygulamalar üzerinden bireysel olarak veya gruplar halinde iletişim kurulabilmektedir. Kullanıcı gruplarında üretilen metin verisini işleyerek anlamlar çıkarmak, sohbetin kalitesini arttırmaktadır. Binlerce kullanıcıya sahip gruplarda birden fazla kez aynı sorunun sorulmasına engel olacak bir sistem oluşturmak, sohbetteki verimliliği artırmaya yardımcı olacaktır. Bu çalışmada Slack platformu üzerinde geliştirilen uygulama sayesinde, kullanıcılar eski mesajlarda benzer soruları arayabilmekte ve cevaplarına ulaşabilmektedir. Sohbette benzer soru bulunmadığı durumlarda sorulan soruyu, konuya hakim kişilere yönlendirecek mekanizma da geliştirilmiştir. Doğal dil işleme tekniklerinden cümle sınıflandırma ve soru cevaplama görevleri için BERTurk modeli kullanılarak ince-ayar yapılmıştır. Yüksek sayıda eğitim verisine ihtiyaç duyulduğundan, Türkçe için eğitim verisi diğer dillerde oluşturulan veri setlerinden çevrilerek ve açık kaynak olarak paylaşılan verisetleri düzenlenerek bu ihtiyaç karşılanmıştır.

Keywords: Doğal Dil İşleme, Derin Öğrenme, Soru Cevaplama, Soru Tanıma

ACKNOWLEDGEMENTS

I would like to thank all my loved ones who supported me during this period, especially my mother, my sister and my friends whom I turned down every plan they made. I would also like to express my gratitude to my dear advisor Assoc. Prof. Derya Karagöz for always supporting me and caring about my opinions, to my dear teacher Res. Asst. Mustafa Murat Arat who introduced me to natural language processing and also to all my professors at Gazi University, METU and Hacettepe University.

I would also like to thank Assoc. Prof. Savaş Yıldırım from Istanbul Bilgi University who supported me to fix the algorithm, to Prof. Dr. Banu Diri from Yıldız Tenik University and Asst. Prof. Zeynep Banu Özger from Kahramanmaraş Sütçü İmam University for providing data that I could use in my research.

CONTENTS

Page

ABSTRACT i

ÖZET ii

ACKNOWLEDGEMENTS iii

CONTENTS iv

TABLES vi

FIGURES vii

ABBREVIATIONS ix

1. Introduction 1

   1.1. Background 2

   1.2. Related Work 3

   1.3. Outline of the Thesis 5

2. Methodology 6

   2.1. Sequence to Sequence Learning 9

   2.2. Transfer Learning 11

   2.3. Transformer Architecture and BERT 12

   2.4. BERTurk 19

   2.5. Downstream NLP Tasks 20

      2.5.1. Sentence Classification 20

      2.5.2. Question Answering 22

      2.5.3. Zero-Shot Classification 23

   2.6. Evaluation Metrics 24

   2.7. Related Technologies 26

3. Application 27

   3.1. Architecture 28

   3.2. Model Preparation 29

      3.2.1. Data Collection 29

      3.2.2. Fine-Tuning 37

3.3. Message Pre-Processing 43
3.4. Production 47
4. Results 51
4.1. Text Classification 51
4.1.1. Dialog Dataset 51
4.1.2. Quora Dataset 55
4.1.3. Tweet Dataset 58
4.2. Question Answering 61
4.2.1. TQuad Dataset 61
4.2.2. YTU QA Dataset 64
5. Conclusion 67
REFERENCES 69
CURRICULUM VITAE 79

TABLES

Page

Table 2.1 Confusion Matrix 24

Table 3.1 Distribution of Dataset Instances for Text Classification 34

Table 3.2 Distribution of Dataset Instances for Question Answering 36

Table 3.3 Hyper-parameter Setup for NLP Tasks 39

Table 4.1 Dialog Dataset Fine Tuning Metrics 55

Table 4.2 Quora Dataset Fine Tuning Metrics 58

Table 4.3 Tweet Dataset Fine Tuning Metrics 61

Table 4.4 TQuad Dataset Fine Tuning Metrics 63

Table 4.5 YTU QA Dataset Fine Tuning Metrics 66

FIGURES

Page

Figure 2.1 Tokenization 8

Figure 2.2 Embedding Vector 9

Figure 2.3 Encoder - Decoder Architecture [56] 10

Figure 2.4 Recurrent Neural Network Structure [19] 11

Figure 2.5 Architecture of Transformer 12

Figure 2.6 Architecture of BERT 17

Figure 2.7 Pre-training Procedure of BERT 19

Figure 2.8 Sentence Classification with BERT 21

Figure 2.9 Question Answering with BERT 22

Figure 3.1 Architecture of the Application 28

Figure 3.2 Dialog Dataset Sample 30

Figure 3.3 Translated Dataset Sample 31

Figure 3.4 Quora Dataset Sample 31

Figure 3.5 Wikipedia Turkish Sentence Dataset Sample 32

Figure 3.6 Quora-Wikipedia Mixed Dataset Sample 32

Figure 3.7 Raw Tweet Question Dataset Sample 33

Figure 3.8 Tweet Question Dataset Sample 34

Figure 3.9 Question Answering Dataset Structure 35

Figure 3.10 PoS Tagging 37

Figure 3.11 PoS Tagging without Tokenization 37

Figure 3.12 PoS Tagging with Tokenization 37

Figure 3.13 Python Code Snippet for Extracting Encodings 39

Figure 3.14 Python Code Snippet for Trainer Class in Text Classification 40

Figure 3.15 Python Code Snippet for Compute Metric Function 41

Figure 3.16 Python Code Snippet for Trainer Class in Question Answering 41

Figure 3.17 Python Code Snippet for Inference in Text Classification 42

Figure 3.18 Python Code Snippet for Inference in Question Answering 43
Figure 3.19 Python Code Snippet of Message Class.................................. 44
Figure 3.20 Message Processing Flow 45
Figure 3.21 Chat Group Conversation Windows...................................... 46
Figure 3.22 Python Code Snippet of Question Class 47
Figure 3.23 Python Code Snippet of Answer Class.................................. 47
Figure 3.24 Initializing Bot 48
Figure 3.25 Asking Question... 48
Figure 3.26 Similar Question Selection 49
Figure 3.27 Displaying Answers... 50
Figure 4.1 Dailog Dataset Character Count Distribution........................... 52
Figure 4.2 Dailog Dataset Word Count Distribution 52
Figure 4.3 Dailog Dataset Pos Tags 53
Figure 4.4 Dialog Dataset Fine Tune Metrics by Step Charts....................... 54
Figure 4.5 Quora Dataset Character Count Distribution 55
Figure 4.6 Quora Dataset Word Count Distribution 56
Figure 4.7 Quora Dataset Pos Tags... 56
Figure 4.8 Quora Dataset Fine Tune Metrics by Step Charts 57
Figure 4.9 Tweet Dataset Character Count Distribution............................ 58
Figure 4.10 Tweet Dataset Word Count Distribution 59
Figure 4.11 Tweet Dataset Pos Tags 59
Figure 4.12 Tweet Dataset Fine Tune Metrics by Step Charts 60
Figure 4.13 TQuad Dataset Character Count Distribution of Answers 62
Figure 4.14 TQuad Dataset Word Count Distribution of Answers.................. 62
Figure 4.15 TQuad Dataset Fine Tune Metrics by Step Charts 63
Figure 4.16 YTU QA Dataset Character Count Distribution of Answers 64
Figure 4.17 YTU QA Dataset Word Count Distribution of Answers 64
Figure 4.18 YTU QA Dataset Fine Tune Metrics by Step Charts.................... 65

ABBREVIATIONS

AI	:	**A**rtificial **I**ntelligence
ANN	:	**A**rtificial **N**eural **N**etwork
API	:	**A**pplication **P**rogramming **I**nterface
BERT	:	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
CRF	:	**C**onditional **R**andom **F**ields
DL	:	**D**eep **L**earning
DNN	:	**D**eep **N**eural **N**etwork
ELMo	:	**E**mbeddings from **L**anguage **Mo**dels
FFNN	:	**F**eed-**F**orward **N**eural **N**etwork
GPT	:	**G**enerative **P**re **T**raining
GPU	:	**G**raphics **P**rocessing **U**nits
HTTP	:	**H**yper-**T**ext **T**ransfer **P**rotocol
JSON	:	**J**avaScript **O**bject from **N**otation
LSTM	:	**L**ong-**S**hort **T**erm **M**emory
ML	:	**M**achine **L**earning
MLM	:	**M**asked **L**anguage **M**odeling
NLP	:	**N**atural **L**anguage **P**rocessing
NSP	:	**N**ext **S**entence **P**redicton
PoS	:	**P**art-**o**f-**S**peech
RNN	:	**R**ecurrent **N**eural **N**etwork
TPU	:	**T**ensor **P**rocessing **U**nits

1. Introduction

Communication is the fundamental necessity of our age. With the global spread of the internet, people can communicate with each other wherever they are in the world. The main instrument of communication is language, and considering that there are more than 7000 languages in the world [82], it becomes critical to develop technology in this area.

Whereas online forums were extensively used when communicating over the internet, nowadays, social media applications such as WhatsApp, Slack, Discord, Telegram, etc. are preferred. People can create conversation groups around any subject using these applications. In addition, experts and beginners could join these groups, such as ”Deep Learning Group”, ”Machine Learning Techniques”, ”Artificial Intelligence Working Group” etc. Considering that there are thousands of people in these groups and that new people continue to join, it is inevitable that the same topics are discussed, and therefore the same questions are asked over and over again. The reason for this situation is that conversation groups have a continuous flow, therefore a message about a topic may disappear in minutes, even in seconds, unlike forums [25]. The fact that machine learning or deep learning techniques are replacing repetitive tasks directly affects this situation. There are various support systems to prevent the same questions from being asked repeatedly by different people. For example, while you are using telephone banking, if you ask ”How much money do I have in my account?” to the voice response system, it gives the answer itself by checking your account instead of connecting you to the customer representative. Another example would be tickets that are opened to the information technology (IT) help desk. Password reset requests can even be easily carried out without any human interaction.

Although certain conditions could be controlled in some industries, such as banking, telecom, IT departments, etc., it is difficult to filter messages that are sent randomly in a conversation groups. The first requirement for such a scenario is an automation system, which is responsible for processing text inputs. Messages could be filtered or processed with bots in the social media applications mentioned above [76]. A bot is a software application that is programmed to perform specific tasks. Bots can perform predefined tasks with the help of commands, as well as send messages to a remote server received from users, allowing them to be processed. In this way, messages can be processed using Natural Language Processing (NLP) techniques and the output can be returned as a response. The second requirement is to analyze these inputs. Since these text inputs are sentences constructed in daily life, they should be translated into machine language. Understanding human language by machines requires a lot of subtasks.

This study mainly focuses on preventing the same questions from being asked in conversation groups, thereby increasing conversation quality. In order to do so, three different NLP tasks have been carried out; (1) sentence classification for detecting questions, (2) question answering for pairing appropriate answers with detected questions, and (3) zero-shot classification for matching the subject of each sentence with users.

1.1. Background

NLP concentrates on how natural language can be understood by machines. It started its journey in the 1950s with hand-written rules. In the following years, statistical methods, machine learning, and finally Deep Learning (DL) algorithms made NLP progress and achieved the state-of-art results.

Computers, unlike humans, can only understand numbers. Therefore, we have to represent words in a numeric format that is understandable by computers. In order to convert a text input to a numerical value, it should be broken into pieces. These pieces are called tokens, and this process is called tokenization. Word embedding comes after this very beginning process. It is used to represent words with numeric vectors.

In the 2010s, Recurrent Neural Networks (RNN) started to be used to process these vectorized text inputs. These networks process sentences one at a time in a concept that is called memory. Since this memory has certain limits, the Long Short-Term Memory (LSTM) model has been proposed to overcome the short memory problem [13].

Another development in NLP is transfer learning. Transfer learning was first published and used in neural networks in 2008 by Collobert et al. [18]. It is used in pre-trained models today. Through transfer learning, the information obtained in the first layers of an artificial neural network can be distributed to be used in different tasks.

By 2015, the attention mechanism had emerged as a groundbreaking technique in NLP, replacing recurrence. As the name suggests, this mechanism concentrates on some words/tokens in the sentence/sequence at each step in order to find the most related part. Thus, the problem of compressing all information into a fixed-size vector in previous models, has been solved in the encoder-decoder part of this mechanism. BERT [22], RoBERTa [48], GPT [59], GPT-2 [60] and GPT-3 [12] models have been proposed by using an attention mechanism, and they continue to work with this architecture, because the attention mechanism is very powerful in finding relationships between words.

1.2. Related Work

As previously mentioned, this study mainly focuses on three different sub-tasks of NLP and the methods that are required to solve them. Therefore, this section consists of previous research conducted on text classification and question answering tasks. Since NLP techniques need to be language-specific, the literature for both English and Turkish has been properly reviewed.

Currently, DL techniques are popular to determine whether a sentence is a question, rather than using rule-based methods. Rule-based systems are concerned with whether or not the sentence contains wh- questions, a question mark (?), or regex-determined patterns. Although a rule-based system produces significant results for the dataset it is working on, it would not work at the same performance for different datasets. For instance, Kwong et al. [45] compared three different algorithms, which are naive, regex, and S&M [74] in order to detect questions. In contrast to rule-based methods such as naive and regex, S&M algorithm uses part-of-speech tags of sentences as a feature set to learn whether a sentence is a question or not. It has been seen that rule-based methods give superior results on a given dataset in the project. Özger et al. [53] obtained a model that reaches approximately 88% accuracy by combining rule-based methods in hand-crafted Turkish tweet dataset. They combined question marks, question affixes, question words, and special words (e.g. acaba, demi, sence) as features and trained their model with conditional random fields (CRF).

Sentences can be categorized by using classification techniques. LSTM models with roughly 96% accuracy have been proposed, which was popular recently, [38] but models that are constructed with BERT architecture are more promising due to higher accuracy [84]. In recent studies, DL techniques have shown better results in question classification. Jiang et al. [39] searched for a solution to classify double-barreled questions, which contain two questions in one sentence. In their approach, they showed that DL algorithms such as BERT, Word2Vec, XLNet gives higher accuracy rather than rule-based systems.

Question-answering, which is considered one of the information retrieval problems in engineering, is a research area defined as a reading comprehension problem in the NLP community [20]. Whereas well-defined datasets were used in previous studies such as Baseball, LUNAR, WOLFIE, TREC-8, today's models are trained with modern datasets such as SQuAD, CNN/Daily mail, TriviaQA, MS MARCO [68, 86].

In the question-answering task, there are two different approaches: knowledge-based and model-based. Knowledge-based systems are utilized in search engines; these systems are made up of structured or unstructured data that require a lot of engineering and are created according to a set of rules. The user's inputs are searched in these knowledge bases, and the most relevant results are displayed. On the other hand, model-based question-answering systems have two inputs: a question and a context. Contexts could be a paragraph or a document, and the question is searched in the context and a short answer is displayed. Since the model-based systems have limitations, the text is divided into parts and answers are sought in these parts in order to find an answer to a question in a long text entry [42]. The improvements in NLP have resulted in the development of models such as LSTM, BERT, GPT, which provide superior outcomes in the question-answering problem [2].

1.3. Outline of the Thesis

The remaining part of this thesis is composed as follows. The evaluation of NLP techniques and related technologies was introduced in Chapter 2. Data collection, the fine-tuning process, and the architecture of the application were explained in Chapter 3. The fine-tuning metrics were given in Chapter 4. Finally, a conclusion was given in Chapter 5.

2. Methodology

Language is generally regarded as the distinguishing feature of human intelligence. As a result, one of the most greatest challenges to the advancement of artificial intelligence is the creation of systems capable of understanding human language. This goal has guided artificial intelligence research, particularly in NLP and computational linguistics. Since language pervades every element of human existence, NLP is required for computers to fulfill their full potential in improving human intelligence. NLP, which is now one of the trending artificial intelligence subjects, has a small number of practical applications and a lot of theoretical research. It is a branch of computer science and linguistics that explores how natural language texts and/or sounds are processed in a computer system [21].

NLP started to be studied in the 1950s, but it was suspended for a while due to a lack of resources. It resurfaced with the development of statistical methods in the 1980s. Studies were based on a set of rules until the 1980s, and with the increase in machine power and labeled data in the 2000s, unsupervised machine learning methods began to be used in this field. Deep neural networks (DNNs) which are deep version of artificial neural networks (ANNs), now provide superior outcomes in many NLP tasks [83]. ANNs are the main algorithms underlying deep learning (DL) [47]. It's at the core of a lot of recent AI applications. DNNs have the ability to handle highly complex and large machine learning tasks due to the fact that they are powerful and versatile. [27]. In recent years, DNNs has also been used in computer vision problems [44, 80], NLP, and even in business and finance applications such as insurance credit scoring [43] and financial market forecasting [26, 34].

Although neural networks were proposed in the 1940s, most of the basic concepts for DL, such as shallow neural networks, were developed in the 1980s [81]. However, it was difficult, almost impossible to train the deep layers of a network due to a lack of resources. Therefore, along with many other factors, the two most important ones play a big role in the deep learning revolution.

The emergence of large volumes of high-quality data sets for training models, as well as the use of affordable data centers to store these data sets, is the first factor in the DL revolution. Working with a large data set with many variables (rather than using handcrafted variables obtained by feature engineering methods) requires large-scale training data because of the size problem [10]. Whereas it was almost impossible to collect data and especially to store these data. DL techniques are now widely used as a result of training models by companies and organizations collecting large amounts of different data. Deep learning technology has progressed significantly, allowing even end users to access these trained models easily (e.g. cloud-based systems).

The second factor is the emergence of powerful, massively parallel computing power with graphics processing units (GPU) [52, 61] and tensor processing units (TPU) [41]. Computation and memory usage increased linearly with the number of observations in the dataset. Since DNNs are both computationally and financially expensive, the demand for computer resources has risen dramatically in order to efficiently complete both training and inference tasks in a reasonable amount of time. Increasing computational power has led to an increase in the capacity of neural networks, in other words, the number of parameters. Neural networks become more useful and significant by increasing the capacity, thus the prediction accuracy has improved [70].

DL is used in many fields such as computer image processing and pattern recognition. With the experience gained in this field, new DL algorithms have started to emerge for NLP. Many techniques used in machine learning have also been used to solve NLP problems. In recent years, dense vector representation neural networks have surpassed classic neural networks in a range of NLP tasks [85].

Since natural language is such an important aspect of our lives, NLP applications have extended widely. Some of these applications are sentence classification, information extraction, semantic parsing, question answering, multi-document summarization, language-to-language translation, speech and character recognition, etc. [51]. While each of these tasks used to be treated as a different topic of study, they are now covered using common models.

The lexical or prosodic features of a sentence might be utilized when performing one of the NLP tasks on it [16]. While prosodic features deal with where the intonation is, structural features take into account the positions of the elements (subject, predicate, etc.) in the sentence. Although prosodic features appear to improve model performance, it is not possible to extract prosodic features for messages in a chat group [49]. Therefore, it is necessary to create a model using structural features and implement the desired NLP tasks.

The very initial step of any NLP task is to extract tokens from sentences. The tokenization process is used to split sentences into smaller pieces. The process may vary according to the selected algorithm. While some algorithms can perform tokenization according to words, some can do it according to sub-words. BERT, which is used in this study, uses WordPiece [71] tokenization method. It is a sub-word tokenization method, and it uses likelihood to split words. For example, if the likelihood of the word ”OT” is higher than the letters ”O” and ”T” separately, the splitting process is not performed. As can be seen in Figure 2.1, words whose meanings change when adding suffixes such as ”Oduncu” and ”Odunluk” preserve their semantic meanings by separating their roots with the WordPiece method.

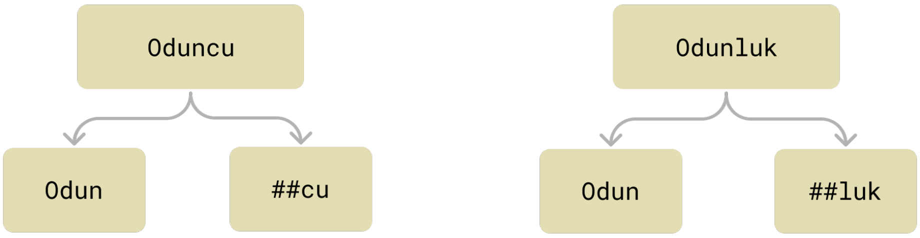
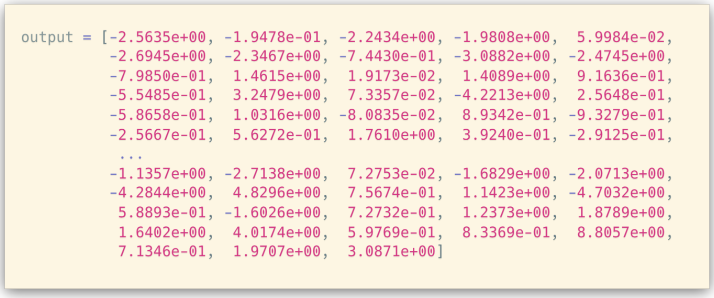


Figure 2.1 Tokenization

Before working on a sentence or word, it is necessary to convert it to numerical values. The word embedding methods are used for creating numerical vectors as can be seen in Figure 2.2 and these vectors are used as the first layer of an ANN.



```
output = [-2.5635e+00, -1.9478e-01, -2.2434e+00, -1.9808e+00,  5.9984e-02,
          -2.6945e+00, -2.3467e+00, -7.4430e-01, -3.0882e+00, -2.4745e+00,
          -7.9850e-01,  1.4615e+00,  1.9173e-02,  1.4089e+00,  9.1636e-01,
          -5.5485e-01,  3.2479e+00,  7.3357e-02, -4.2213e+00,  2.5648e-01,
          -5.8658e-01,  1.0316e+00, -8.0835e-02,  8.9342e-01, -9.3279e-01,
          -2.5667e-01,  5.6272e-01,  1.7610e+00,  3.9240e-01, -2.9125e-01,
           ...
          -1.1357e+00, -2.7138e+00,  7.2753e-02, -1.6829e+00, -2.0713e+00,
          -4.2844e+00,  4.8296e+00,  7.5674e-01,  1.1423e+00, -4.7032e+00,
           5.8893e-01, -1.6026e+00,  7.2732e-01,  1.2373e+00,  1.8789e+00,
           1.6402e+00,  4.0174e+00,  5.9769e-01,  8.3369e-01,  8.8057e+00,
           7.1346e-01,  1.9707e+00,  3.0871e+00]
```

Figure 2.2 Embedding Vector

The most commonly used technique, named Word2Vec [50], contains general information about the given sentence in the vector, but it is not able to contain the properties of the sentence complementation, polysemy, anaphora, harmony, negation, etc. [66]. At this point, in 2018, which is also seen as the golden age of NLP, a technique called ELMo (Embeddings From Language Models) was developed in order to solve the polysemy problem [54].

The benefits of the pre-training mechanism have also emerged with this technique based on the bidirectional LSTM architecture [64]. Pre-training is the technique of using the parameters (weights) collected from a trained model for a particular task as initial values for another task.

2.1. Sequence to Sequence Learning

The technique that reads sentences sequentially and maps one sequence to another is known as ”sequence-to-sequence learning”. There is an encoder-decoder architecture in this learning technique. The encoder reads the input sequence and creates a vector including information about the sentence, which is then given to the decoder as an input. Longer sentences would cause a loss of information in the output vector that the encoder produces, which is known as ”long-term dependency”.

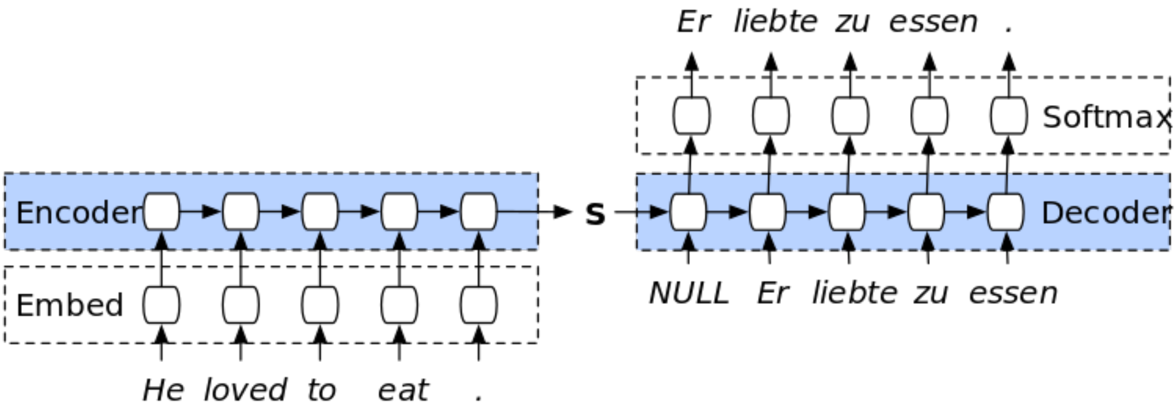


Figure 2.3 Encoder - Decoder Architecture [56]

Figure 2.3 describes the encoder-decoder architecture in sequence-to-sequence models. This architecture was first introduced by Cho et al. [15] but due to a long-term dependency problem, Sutskever et al. [78] used an LSTM-based encoder-decoder model. Although LSTM networks are used to tackle long-term dependency, training of these type of deep networks is still a problem.

Languages have a sequential structure by their nature. One of the requirements of this structure is that the information at the beginning of the sequence and another piece of information at the end can be related to each other. For example, ”I have a black cat. That’s why I brush its fur every day.” In the sentence, the words ”cat” and ”fur” are related. RNNs are used in such case because sequential structure cannot be modeled with classical machine learning techniques. RNNs process one token at a time and have a chain structure that transfers the obtained information to the next cell. As can be seen in the RNN structure which is given in Figure 2.4, O_{t-1} obtained from X_{t-1} is given as input to the next cell together with X_t.

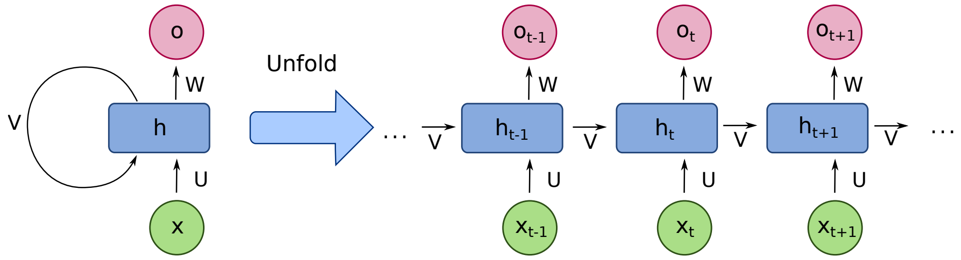


Figure 2.4 Recurrent Neural Network Structure [19]

The disadvantage here is that the context cannot be tracked in cases where the gap between the tokens is too wide. As a result, the information between the words ”cat” and ”fur,” given in the example sentence above, can be lost. RNNs takes whole sequence and does not concern whether the given tokens of sequences is important. LSTM cells, on the other hand, can remember important information with a cell state mechanism. But we face the same problem here: as the length of the sequence increases, the important information will be lost. Another problem both network cannot process the sequence in parallel. Therefore they require huge computational power.

2.2. Transfer Learning

As mentioned above, an NLP project includes more than one task. In the past, it was necessary to produce different models for each task and train each model from scratch. The training process is very time-consuming and requires a huge amount of computational power. Transfer learning also known as domain adaptation, which emerged as a solution to this situation, allows pre-trained models to be fine-tuned. Fine-tuning is the key process for adapting the pre-trained model to new tasks.

In 2008, the word embedding layer was shared between different models with the multi-task learning technique [18, 67]. In 2017s, transfer learning was applied to NLP tasks, which has already used in computer vision problems [79]. Transfer learning makes it possible to use the information obtained in the upper layers of a model by transferring it to different models.

Instead of creating new models for different NLP tasks, ULMFiT (Universal Language Model Fine-tuning) uses the transfer learning architecture as its basis, and it is able to use the main information created in a particular model in other tasks [33].

2.3. Transformer Architecture and BERT

Vaswani et al. [79] introduced a revolutionary work in NLP in 2017. With the ’Attention’ mechanism they suggested, they found solutions to many problems encountered, as well as speeding up the training process and causing high performance models to emerge.

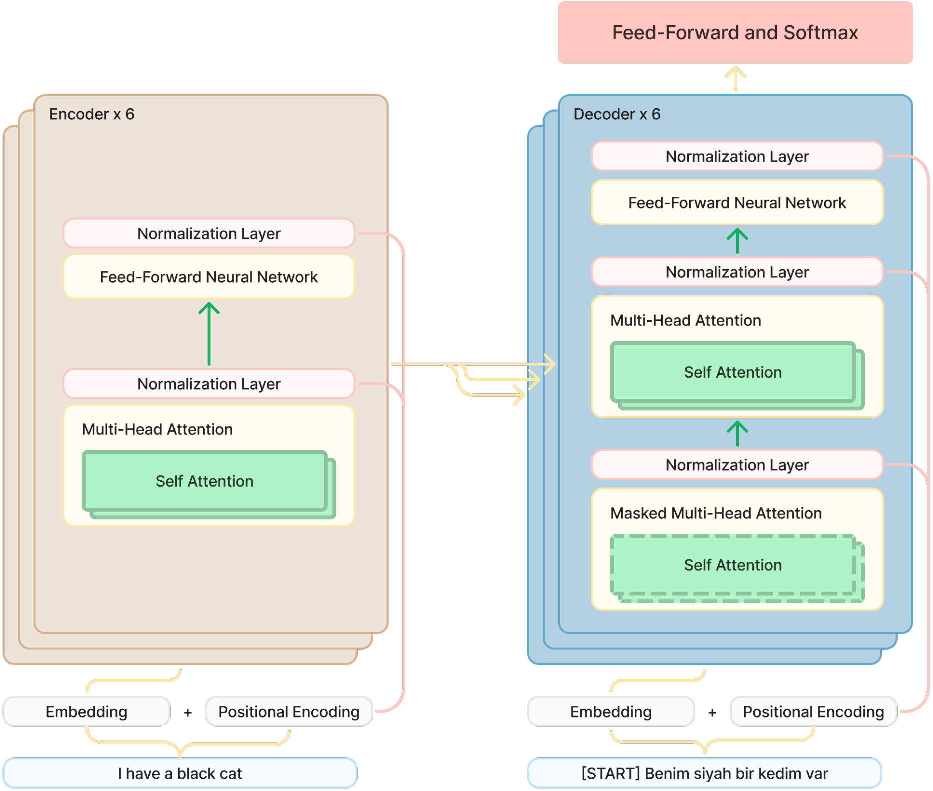


Figure 2.5 Architecture of Transformer

The original transformers, in which the entire network structure is visualized in Figure 2.5, have an encoder-decoder architecture. In the architecture with 6-layer encoder stack on the left and 6-layer decoder stack on the right, the data processing starts with the extraction of the embedding and positional encoding of the sentences. Unlike sequential models, Transformers are fed with a whole sentence at once, therefore the position information of the tokens is kept in the positional encoding vector.

The most important feature of this model is the attention mechanism. The attention mechanism which exists in both the encoder and the decoder, numerically represents the relationship between the words of the sentences. The idea behind attention is that the network should figure out which parts (elements) of the input sequence are more relevant to generate a given element of the output sequence. The multi-head module contains more than one self-attention layer which is determined as 8 in the original paper. Thus, more information is obtained by focusing on the different features of the words in each layer. The calculation of self-attention score is given as follows:

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{Z}^i = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\mathbf{V}, \quad \text{for} \quad i = 1, ..., 8. \tag{1}$$

Here, $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ stand for Query, Key, and Value matrix, respectively. These concepts work similarly to information retrieval systems [24]. $\mathbf{Q}$ can be thought of as an input token, $\mathbf{K}$ is the searched sequence, and $\mathbf{V}$ is the corresponding output. d_k is a constant which is the dimension of $\mathbf{K}$ and it is specified as 64 in the original paper. i is the index of self-attention layer. Calculation of the Query, Key, and Value matrix, respectively, is given below:

$$\mathbf{Q} = \mathbf{X} \times \mathbf{W}^Q, \quad \mathbf{K} = \mathbf{X} \times \mathbf{W}^K, \quad \mathbf{V} = \mathbf{X} \times \mathbf{W}^V \tag{2}$$

where $\mathbf{X}$ is the input matrix and $\mathbf{W}$s are initial weights which are trainable parameters.

Softmax is a function that generates a probability distribution from an input vector [28]. This function is used in multi-class classification problems. The probability distribution can be obtained with the following equation:

$$\mathrm{softmax}(\vec{z})_i = \frac{e^{z_i}}{\sum\limits_{j=1}^{K} e^{z_j}}, \quad \text{for} \quad i = 1, ..., K \tag{3}$$

where $\vec{z} = z_1, ..., z_K$ is the input vector, z_i is the ith element of the input vector, K is the number of classes e is the standard exponential function. By using Eq. (3), it normalizes the input sequence and ensures the sum of the output vector is 1.

Encoder Attention Layer

Considering the ”I have a black cat” sequence, the size of the input tokens is 5. The size of the input embeddings for each token is 512 which is determined in the original paper. Therefore, input $\mathbf{X}$ would be a 5×512 dimensional matrix. Matrix dimensions of Eq. (2) can be found below:

$$\begin{aligned}
\mathbf{Q}_{5\times 64} &= \mathbf{X}_{5\times 512} \times \mathbf{W}^{Q}_{512\times 64} \\
\mathbf{K}_{5\times 64} &= \mathbf{X}_{5\times 512} \times \mathbf{W}^{K}_{512\times 64} \\
\mathbf{V}_{5\times 64} &= \mathbf{X}_{5\times 512} \times \mathbf{W}^{V}_{512\times 64}
\end{aligned} \tag{4}$$

As a result of Eq. (1), $\mathbf{Z}^{i}_{5\times 64}$ matrix is obtained for each self-attention layer. As we mentioned above, there are 8 self-attention layers in the multi-head attention module. The calculation of output matrix of multi-head attention module is given as follows:

$$\mathbf{Z_{5\times 512}} = \begin{bmatrix} \mathbf{Z^1_{5\times 64}} & \mathbf{Z^2_{5\times 64}} & \cdots & \mathbf{Z^8_{5\times 64}} \end{bmatrix}_{5\times 512} \quad \times \quad \mathbf{W_{512\times 512}} \tag{5}$$

Here, each $\mathbf{Z}^{i}_{5\times 64}$ matrix is concatenated and then multiplied with an additional weight matrix, $\mathbf{W}_{512\times 512}$.

The information extracted in the encoder layers is sent to the next encoder layer. There could be more than one encoder layer in order to increase predicting power by extracting different attention representations. The information from the last encoder layer is transferred to the decoders together with the embedding and positional encoding information obtained from the target input.

Decoder Attention Layer

The multi-head attention module in the middle of each decoder layer receives the important information from the encoder. The input sequence of the decoder layer is *"[START] Benim siyah bir kedim var"* which has 6 tokens with a special *[START]* token. The *[START]* token is added, since the decoder layer is autoregressive. The calculation of $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$, respectively, for the decoder layer is given as follows:

$$\begin{aligned}
\mathbf{Q}_{6\times 64} &= \mathbf{X}_{6\times 512} \times \mathbf{W}^{Q}_{512\times 64} \\
\mathbf{K}_{5\times 64} &= \mathbf{Z}_{5\times 512} \times \mathbf{W}^{K}_{512\times 64} \\
\mathbf{V}_{5\times 64} &= \mathbf{Z}_{5\times 512} \times \mathbf{W}^{V}_{512\times 64}
\end{aligned} \tag{6}$$

In the Eq. (6), $\mathbf{X}_{6\times 512}$ matrix is created from the target sequence, which is *"[START] Benim siyah bir kedim var"* and $\mathbf{Z}_{5\times 512}$ comes from the last encoder layer.

There is a masked multi-head attention module before multi-head attention, and it is special for the decoder layer. The Attention mechanism processes the words in the given sentence with each other. This masking process is done in the masked multi-head attention module in order to prevent a word from being attended to the future word. The structure of masked matrix can be found below:

$$\mathbf{M} = \begin{array}{c c} & \begin{array}{ccccc} I & have & a & black & cat \end{array} \\ \begin{array}{c} I \\ have \\ a \\ black \\ cat \end{array} & \begin{bmatrix} 0 & -\infty & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty & -\infty \\ 0 & 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{7}$$

Here, mask matrix M, consist of zeros and negative infinitives. The self-attention formula, which is given in Eq. (1), combined with the masked matrix, is given below:

$$\text{MaskedAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\mathbf{M} + \frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\mathbf{V} \tag{8}$$

When the softmax function is applied to the sum of masked matrix M and the multiplication of query and key matrix, the future words are assigned a value of 0.

In all these matrix operations, there is a normalization layer in front of each module in order to transfer the information between the layers stably and to decrease the training time [8]. There is also a Feed-Forward Neural Network (FFNN) in every pair of encoder-decoder layers which adds linearity to the output matrix. In the last step, probabilities are calculated with the softmax function [6, 55, 65].

By June 2018, a technique called GPT (Generative Pre-Training) emerged, which expands the techniques used in ELMo and ULMFiT. This technique, developed by OpenAI, enables both faster training and detection of complex patterns in sentences as a result of the transition from LSTM-based architecture to a Transformer-based architecture [58, 64]. Following that, the Bidirectional Encoder Representations from Transformers which is known as BERT model was developed, which proved to be extremely successful in a variety of NLP applications. With the help of the ability to handle a sentence from two directions (bidirectional), BERT achieved better results in language modeling compared to previous models [22, 32].

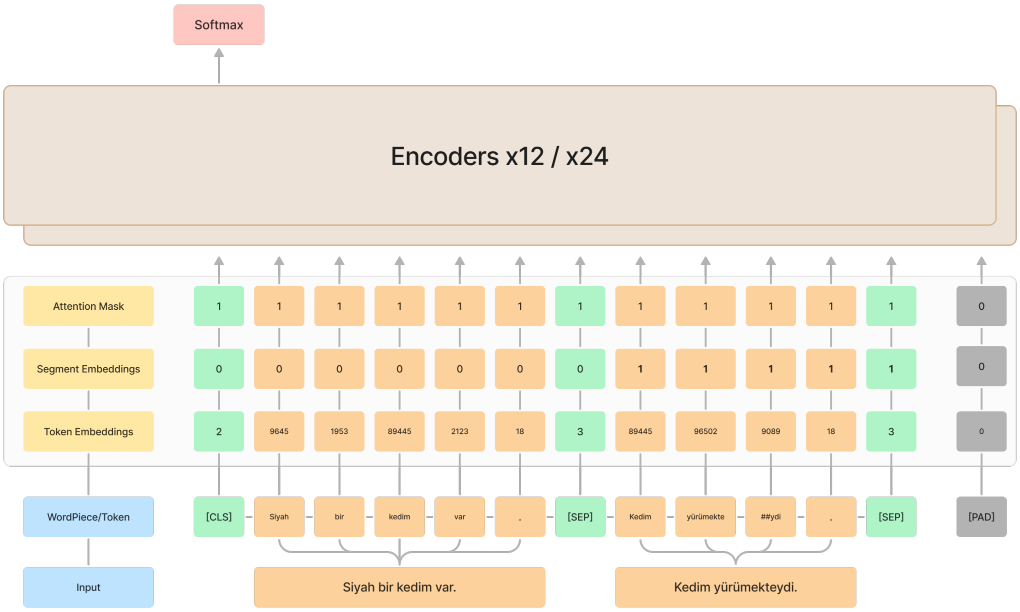


Figure 2.6 Architecture of BERT

BERT is a device that generates vectors with a length of 768 units for each token. It has two different versions, one with 12 layers and the other one with 24 layers and it uses only encoder layer from the Transformer architecture. The overall architecture is visualized in Figure 2.6.

Firstly, the tokens of the input sequence are created. BERT uses the WordPiece method to create tokens. There are also three special tokens: [CLS], [SEP], and [PAD]. The [CLS] token is used for a classification task. The [SEP] token separates two input sequences, and the [PAD] token adjusts the length of the input vectors.

Padding is basically used to adjust vector lengths. The vectors are made to the same length by adding the special [PAD] token that can be seen as follows:

$$\begin{bmatrix} \text{[CLS]} & \text{Siyah} & \text{bir} & \text{kedim} & \text{var} & . & \text{[SEP]} \\ \text{[CLS]} & \text{Beyaz} & \text{köpek} & \text{yürüyor} & \text{[SEP]} & & \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} \text{[CLS]} & \text{Siyah} & \text{bir} & \text{kedim} & \text{var} & . & \text{[SEP]} \\ \text{[CLS]} & \text{Beyaz} & \text{köpek} & \text{yürüyor} & \text{[SEP]} & \textbf{[PAD]} & \textbf{[PAD]} \end{bmatrix} \tag{10}$$

The next step is creating an embedding matrix, which consists of token embeddings, segment embeddings, position embeddings, and attention masks. Token embeddings, or token ids, are the numerical representation of each token. Segment embeddings are used to show which sentence the tokens belong to when there is more than one sentence in the input sequence. Position embeddings indicate the position of each token and attention mask is created to distinguish which token to use in the calculation of attention scores.

BERT is trained with two different tasks, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In the MLM task, 15% of the input is masked, and the goal is to predict these tokens in the softmax layer. 80% of this 15% part is replaced with the [MASK] token, 10% with a random word, 10% remains unchanged. Question answering or natural language inference tasks are focused on understanding the relationships between two sentences. In the NSP task that is run in parallel, 50% of the sequences that are complements to each other in the training set are selected and labeled as `IsNext`. The remaining part is labeled `NotNext` and the training process is completed.

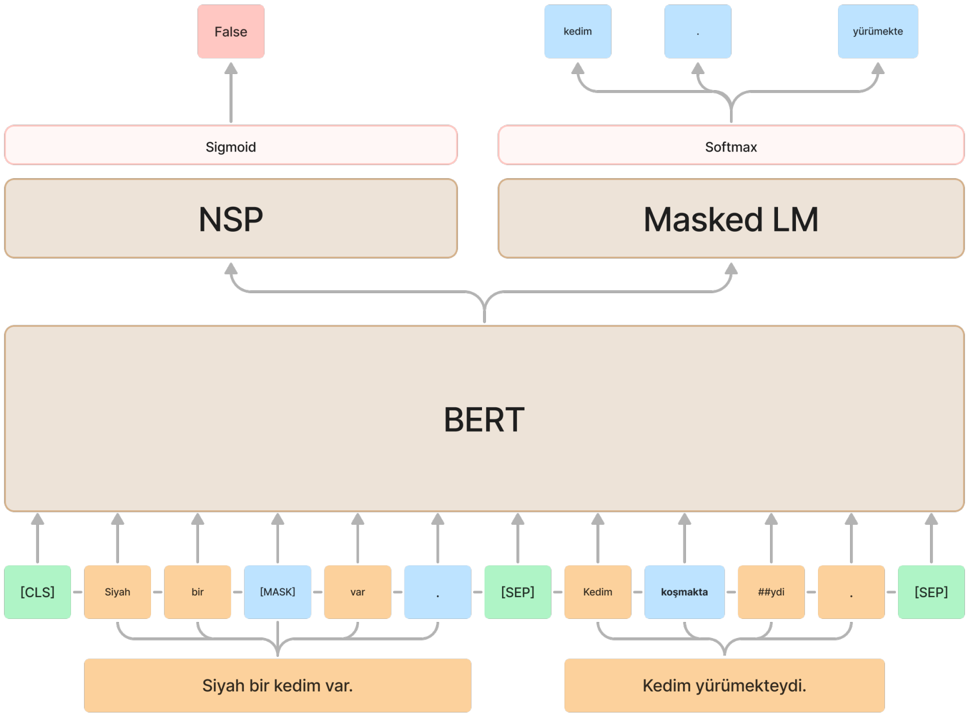


Figure 2.7 Pre-training Procedure of BERT

Figure 2.7 explains the training process of BERT. With the use of this pre-trained model, output vectors can be created for input sequences. These vectors are used as features for subtasks such as text classification or question answering [7, 22].

2.4. BERTurk

The BERT model can be trained for any language when enough raw data is available. BERT has also been trained for Turkish with the data that has been provided by Kemal Oflazer and open source multilingual aggregated data (e.g. Oscar) [72]. Four different models, which were used in our study, are explained as follows.

- **BERTurk 128k:** It is the base model trained for Turkish with the technique described in Section 2.3.
- **DistilBERTurk:** It is a fast and easy-to-use version of the BERT model that is 40% smaller and maintains the accuracy of 97% [69].
- **ConvBERTurk:** Self attentions in BERT encoders consider all context. Jiang et al. [40] proposed a novel method that replaces these attention heads with span based dynamic convolution in order to consider local context.
- **ELECTRA:** The MLM technique works by predicting the originals of tokens that have been replaced by [MASK] tokens. In the method proposed by Clark et al. [17], the input tokens are replaced with other randomly chosen tokens. Then, instead of predicting the original state of the tokens, the classification problem of whether the tokens have been changed or not is examined.

2.5. Downstream NLP Tasks

Transfer learning allows pre-trained models to become widespread, which made it unnecessary to train the models repeatedly for each NLP task. Pre-trained models are trained on high-volume corpora; thus, they can be used in named entity recognition, question answering, sentence classification, etc. only by fine-tuning. Fine-tuning can be defined as using the weights obtained from these pre-trained models as the starting weights for the target task. Thus, superior results are obtained for small datasets, and time and resources can be saved. In this section, how BERT is applied to text classification and question answering tasks and the mechanism of zero-shot classification were explained.

2.5.1. Sentence Classification

Text classification, as in NSP, deals with whether two inputs (sequences) are similar to each other or belong to given classes.

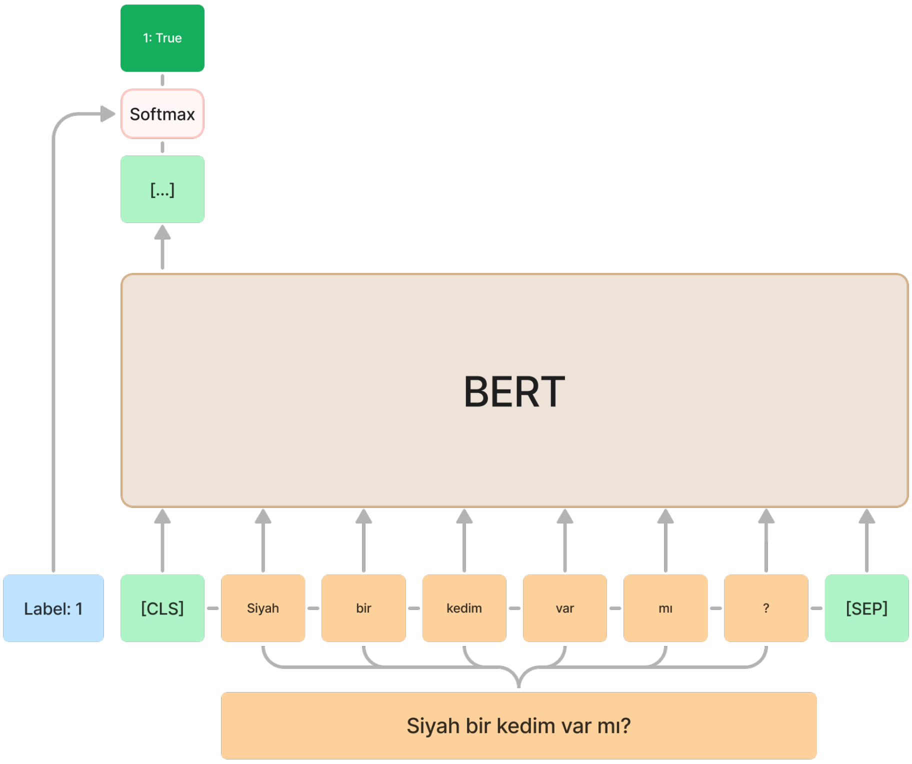


Figure 2.8 Sentence Classification with BERT

As can be seen in Figure 2.8, the process starts with extracting the input tokens and the embeddings obtained from them. The embeddings are passed through 12 layers to create an output vector for each token. All information from the context is collected in the output vector of the special [CLS] token. This vector, with a size of 768, and class labels are given to the softmax function, and output probabilities are produced.

2.5.2. Question Answering

The goal of this task, also known as extractive question answering, is to extract answers that are asked about a paragraph. By expanding the task of predicting the similarity of two sentences to each other, the information about where the answer begins and where it ends is searched in the paragraph.

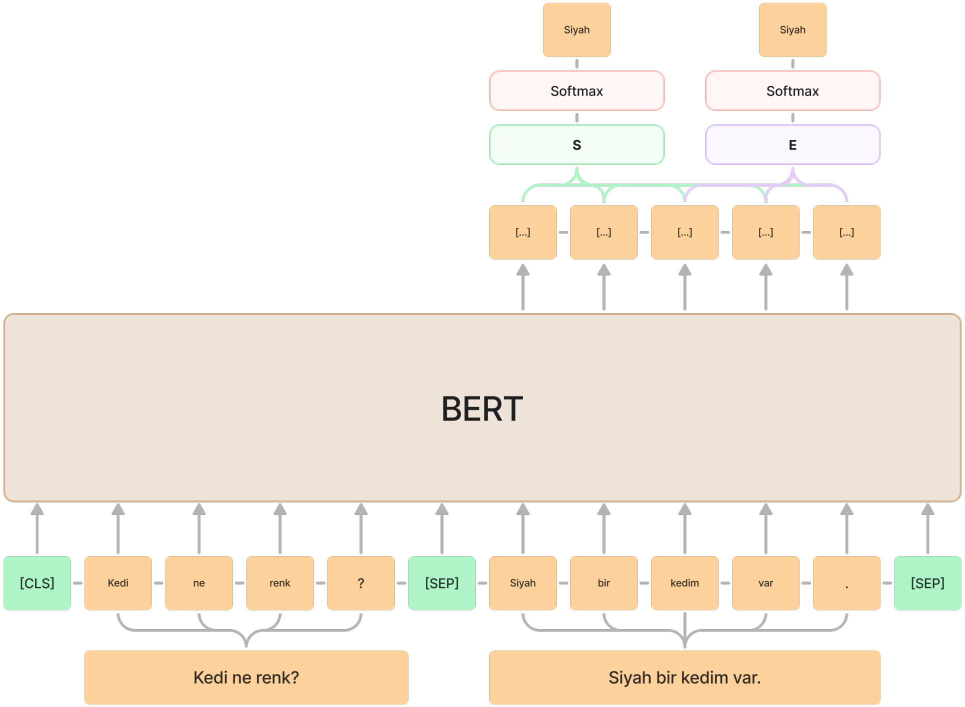


Figure 2.9 Question Answering with BERT

The question on the left in the Figure 2.9 and paragraph on the right are separated with the [SEP] token and the process starts with creating their embeddings. After generating output vectors with BERT, the following equation is used to calculate the probabilities of the start and end tokens of the answer to the question.

$$P_i^S = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}} \qquad P_i^E = \frac{e^{E \cdot T_i}}{\sum_j e^{E \cdot T_j}} \tag{11}$$

where P_i^S and P_i^E denote probability of start and end token, respectively. T_i is the output vector of ith word obtained from the paragraph tokens, e stands for exponential function, S and E represent vector of start and end tokens, respectively. Softmax function is applied to the Eq. (11) to calculate probabilities over the dot product of T_i and the start and end token vectors S and E. The tokens with the highest probability are chosen as the start and end tokens.

In the question-answering task the [CLS] token is used to check if the answer is within the paragraph. In addition, BERT can process sequences with a length of 512 tokens at once due to its nature. The overlapping window technique is used to overcome this difficulty. Paragraphs which are exceeding 512 tokens are cut to the appropriate length, and a new sequence is created so that the end of the cut part and the beginning of the remaining part overlap, and the processes are repeated [73].

2.5.3. Zero-Shot Classification

Most models today require labeled, high-quality and high-volume data. Zero-Shot learning aims to eliminate this necessity. Zero-Shot classification, an unsupervised learning method, has developed models that can achieve superior success in downstream NLP tasks without the need for labeled data. The GPT-3 model, which uses few-shot classification that works with very few labeled data, is one of them. However, too many model parameters require high computational power [12].

Zero-shot classification technique were used in this study to extract sentence categories. The `MoritzLaurer/mDeBERTa-v3-base-mnli-xnli` model on the Huggingface model hub supports 15 languages, including Turkish. Since the model is pre-trained, it can be used directly without the need for fine-tuning [31].

2.6. Evaluation Metrics

It is important to measure the performance of the models during both the training and fine-tuning stages, to see the progress and to determine the best performing model. Different NLP tasks require different performance measurement metrics. For example, Accuracy, Precision, Recall and F1 which are calculated by using the Confusion Matrix given in Table 2.1 are used for sentence classification, while F1 and EM metrics are used for question answering.

		Actual Values	
		Positive (P)	Negative (N)
Predicted Values	Positives	True Positive (TP)	False Positive (FP)
	Negatives	False Negative (FN)	True Negative (TN)

Table 2.1 Confusion Matrix

Accuracy

Accuracy is obtained by dividing correctly classified samples by actual values. It is a basic metric used to measure model performances. It can be calculated with the following equation:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}} \tag{12}$$

where TP stands for true positive, which means the actual value is positive and it is predicted as positive, TN is true negative, which means the actual value is negative and it is predicted as negative, P and N denote the total of the actual positive and negative values, respectively.

Precision and Recall

Precision is obtained by dividing correctly classified samples into all positively classified samples. Recall is a metric obtained by dividing correctly classified samples by actually positive samples. The calculation of Precision and Recall is given below, respectively:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{13}$$

Here, TP is true positive, FP is false positive, which means the actual value is negative and it is predicted as positive, and FN is false negative, which means the actual value is positive and it is predicted as negative.

F1

F1 is used to measure the performance of both classification and question answering models. It is obtained as follows:

$$\text{F1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

In classification problems, although Accuracy gives information about the performance of the models, the F1 score gives more reliable results for unbalanced datasets. F1 is basically harmonic mean of precision and recall.

In question answering problems, when calculating the F1 score, it is concerned with how much the estimated answer and the actual answer cover each other. The F1 formula remains the same, but the Precision and Recall formulas are calculated as follows:

$$\text{Precision} = \frac{\text{Length of Common Tokens}}{\text{Length of Predicted Tokens}} \quad \text{Recall} = \frac{\text{Length of Common Tokens}}{\text{Length of Actual Tokens}} \tag{15}$$

Exact Match

As the name suggests, Exact Match is a metric that deals with whether the estimated response is the same as the actual response. Takes the value 1 if the predicted answer is exactly equal to the correct answer, 0 otherwise.

2.7. Related Technologies

Our work is not the first and it is not going to be last. The increase in human interaction on the internet has made it mandatory for every platform that produces natural language content to integrate their NLP applications. Some platforms provide the data they produce to researchers and enable them to work on it [57, 75] while some platforms serves NLP technology as a service, which they create established with the size of the data they produce [29].

E-commerce sites that interact with people, online banking systems, human resources departments of companies, etc. needed to automate their work as the number of users increases. DeepPavlov [3], which is one of the services that enables this interaction to be made higher quality, offers a conversation framework. With the help of this open source system, entity recognition, intent classification, insult detection can be done in English and Russian. With another service called Talla [1], it is aimed to automate the questions received from the customers by addressing the applications of the companies that will use the service.

3. Application

This chapter consists of four main stages. In the first stage, architecture of the application was explained, in the next stage data collection and fine-tuning process for the downstream tasks was given. In the third stage, processing of the messages in the chat group with fine-tuned models was described. In the last stage, the necessary structures was explained for the bot that the end user would interact with.

In this study, subtasks of NLP called text classification, question answering, and zero-shot classification were used. Roles of these task described as follows;

- **Text Classification:** This task is normally used to categorize sentences. In this study, it was customized to make binary classification to understand whether a sentence is a question or not.
- **Question Answering:** Question answering systems work by extracting short answers from a document or paragraph. It was used to find the proper answer from conversation windows, which is described in section 3.2..
- **Zero Shot Classification:** Apart from other methods, zero-shot classification is based on hypothesis testing. It constructs hypotheses on user-defined classes and distributes them over given labels. Since it is an unsupervised classification technique, there is no need to fine tune it.

The BERT model, which is based on transformer technology, achieved significant results in these tasks. This model requires large volumes of data, as does every deep learning model, but, since it is a pre-trained model, it can be customized for the desired NLP tasks only by fine-tuning. Fine-tuning involves retraining the base model for a specific task. Therefore, high quality labeled data is needed. It is quite difficult to find labeled data for the Turkish language. For this reason, the labeled data that is prepared in different languages has been translated into Turkish with the help of translation APIs, and fine-tuning was applied to the models. All necessary structures, including architecture, data collection and processing, and fine-tuning processes, will be covered in the next sections.

3.1. Architecture

Almost every social media application has API integration. Slack is one of the social media application with the most comprehensive API services. Thus, it was used to create a bot in this study. Also, this application is the key communication tool for the tech industry. It provides tons of features such as connectors to use third party applications (e.g. Outlook, Google Drive, Jira), automation, security, etc. By means of these features, it becomes attractive for use in departments such as human resources, customer relations, marketing, and sales.

Slack API communicates with applications over HTTP calls. The provided endpoints were used to read messages, retrieve user and group information, and deliver the necessary answers to the user. The high-level architecture of the application was given in the Figure 3.1.

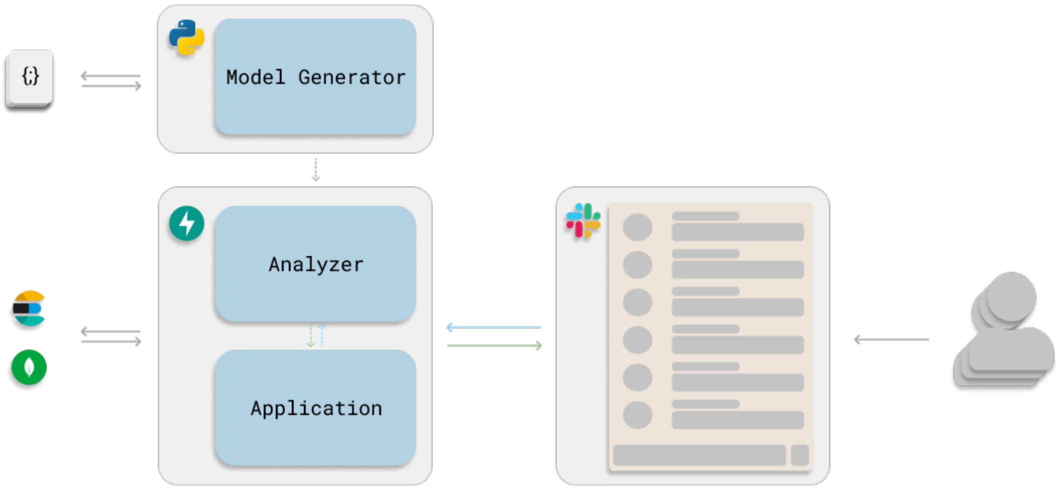


Figure 3.1 Architecture of the Application

As the name implies, FastAPI is a fast, easy to use Python web framework. Analyzer and application module were built on top of this framework. Due to Slack providing a Python client library, it is easy to consume necessary APIs. The function of modules is explained as follows:

- **Model Generator:** There are three different NLP tasks, which are sentence classification, question answering, and zero-shot classification. Data collection and fine-tuning processes for these tasks were held in this module. This module is independent of the main application, therefore it was only used to create models.
- **Analyzer:** This module is responsible for pre-processing messages and analyzing text inputs retrieved from Application module. It was designed to work independently so that by just changing the social media adaptor within the Application module, it can process the text inputs.
- **Application:** This module is an application that acts as a bridge by communicating with the endpoints provided by Slack. It is enough to change the adaptor of this module in order to communicate with other social media applications within the general architecture.

In addition, MongoDB and Elasticsearch were positioned as databases. MongoDB was used to store messages, questions, and answers while vector representations of questions were stored in Elasticsearch.

3.2. Model Preparation

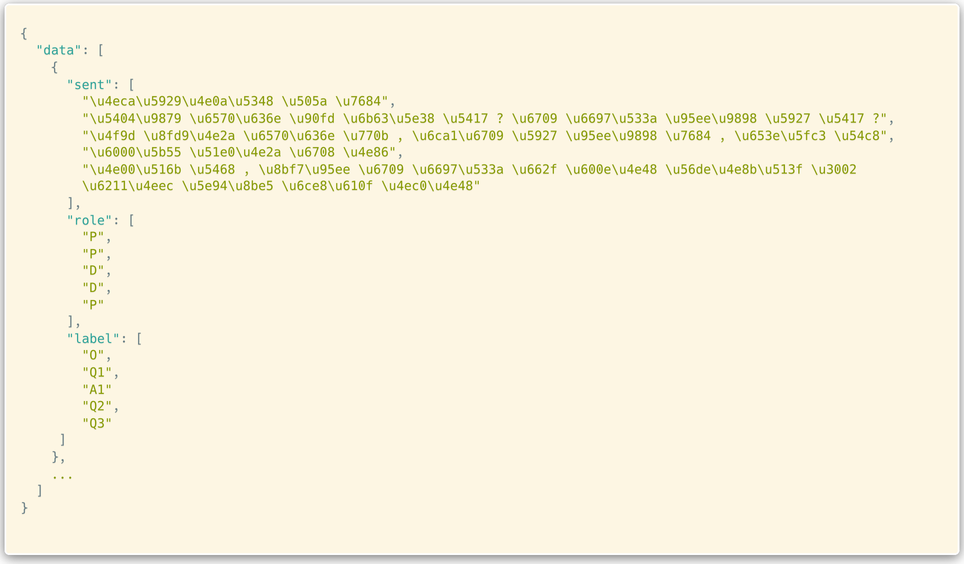
In order to fine-tune the models and perform visual analysis, the datasets must go through a series of processes. In this section, the process of obtaining, preparing, and extracting the grammatical features of the datasets is discussed.

3.2.1. Data Collection

One of the most important steps before fine-tuning is to find a labeled dataset. Due to lack of labeled Turkish datasets, we proposed using datasets in different languages by translating them. We gathered three different datasets for text classification and two different datasets for question answering. Dataset structures and contents were explained as follows.

Text Classification

The first dataset is the doctor-patient dialog dataset in Chinese published by Jia et al. [37]. Dataset consists of 1000 dialog sessions and is separated into 700 training, 200 test, and 100 validation sets in JSON format.



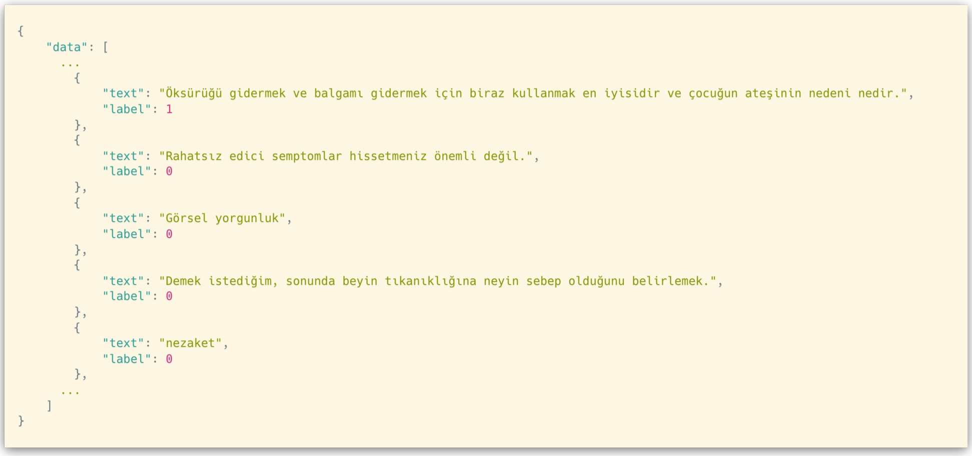
```
{
  "data": [
    {
      "sent": [
        "\u4eca\u5929\u4e0a\u5348 \u505a \u7684",
        "\u5404\u9879 \u6570\u636e \u90fd \u6b63\u5e38 \u5417 ? \u6709 \u6697\u533a \u95ee\u9898 \u5927 \u5417 ?",
        "\u4f9d \u8fd9\u4e2a \u6570\u636e \u770b , \u6ca1\u6709 \u5927 \u95ee\u9898 \u7684 , \u653e\u5fc3 \u54c8",
        "\u6000\u5b55 \u51e0\u4e2a \u6708 \u4e86",
        "\u4e00\u516b \u5468 , \u8bf7\u95ee \u6709 \u6697\u533a \u662f \u600e\u4e48 \u56de\u4e8b\u513f \u3002
        \u6211\u4eec \u5e94\u8be5 \u6ce8\u610f \u4ec0\u4e48"
      ],
      "role": [
        "P",
        "P",
        "D",
        "D",
        "P"
      ],
      "label": [
        "O",
        "Q1",
        "A1"
        "Q2",
        "Q3"
      ]
    },
    ...
  ]
}
```

Figure 3.2 Dialog Dataset Sample

The dataset structure was given in Figure 3.2 and it has three elements:

- **sent**: Unicode characters of dialog sentences.
- **role**: The role of the person who wrote the sentence. Patient or doctor.
- **label**: The type of the sentence. Question, answer, or no label.

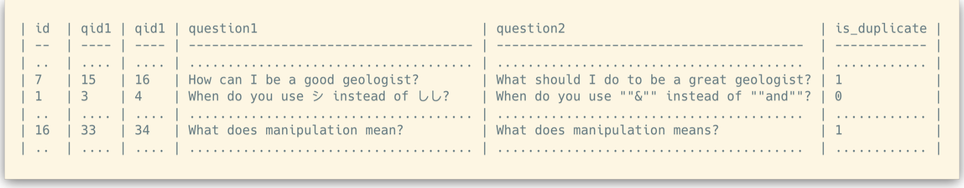
In order to translate dataset Google Translate API has used [30]. The API employs its own translation model, Google Neural Machine Translation (GNMT) [14]. As shown in Figure 3.3, each sentence was translated into Turkish and converted into sentence-based blocks with their label (Question or Answer).



```
{
    "data": [
      ...
        {
            "text": "Öksürüğü gidermek ve balgamı gidermek için biraz kullanmak en iyisidir ve çocuğun ateşinin nedeni nedir.",
            "label": 1
        },
        {
            "text": "Rahatsız edici semptomlar hissetmeniz önemli değil.",
            "label": 0
        },
        {
            "text": "Görsel yorgunluk",
            "label": 0
        },
        {
            "text": "Demek istediğim, sonunda beyin tıkanıklığına neyin sebep olduğunu belirlemek.",
            "label": 0
        },
        {
            "text": "nezaket",
            "label": 0
        },
      ...
    ]
}
```

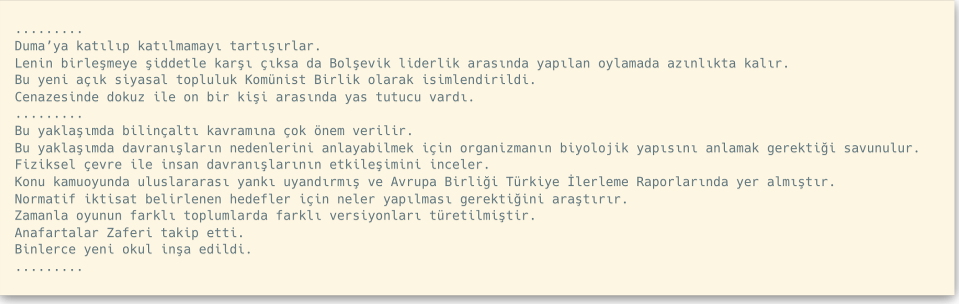
Figure 3.3 Translated Dataset Sample

The second one consists of a combination of Quora [57] and Wikipedia [4] datasets. Quora is a big and versatile question-answering website. They published this dataset to find a solution to duplicate questions. The dataset has three main elements, which can be seen in Figure 3.4. Two columns contain question pairs, and one column indicates whether they are duplicated or not.



```
| id  | qid1 | qid1 | question1                            | question2                                 | is_duplicate |
| --  | ---- | ---- | ------------------------------------ | ----------------------------------------  | ------------ |
| ..  | .... | .... | .................................... | ........................................  | ............ |
| 7   | 15   | 16   | How can I be a good geologist?       | What should I do to be a great geologist? | 1            |
| 1   | 3    | 4    | When do you use シ instead of しし?   | When do you use ""&"" instead of ""and""? | 0            |
| ..  | .... | .... | .................................... | ........................................  | ............ |
| 16  | 33   | 34   | What does manipulation mean?         | What does manipulation means?             | 1            |
| ..  | .... | .... | .................................... | ........................................  | ............ |
```

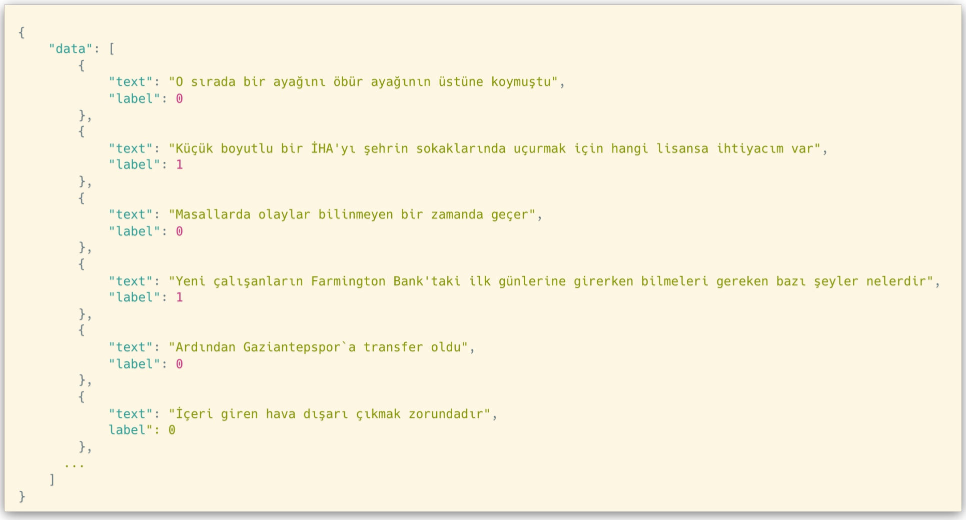
Figure 3.4 Quora Dataset Sample



```
.........
Duma’ya katılıp katılmamayı tartışırlar.
Lenin birleşmeye şiddetle karşı çıksa da Bolşevik liderlik arasında yapılan oylamada azınlıkta kalır.
Bu yeni açık siyasal topluluk Komünist Birlik olarak isimlendirildi.
Cenazesinde dokuz ile on bir kişi arasında yas tutucu vardı.
.........
Bu yaklaşımda bilinçaltı kavramına çok önem verilir.
Bu yaklaşımda davranışların nedenlerini anlayabilmek için organizmanın biyolojik yapısını anlamak gerektiği savunulur.
Fiziksel çevre ile insan davranışlarının etkileşimini inceler.
Konu kamuoyunda uluslararası yankı uyandırmış ve Avrupa Birliği Türkiye İlerleme Raporlarında yer almıştır.
Normatif iktisat belirlenen hedefler için neler yapılması gerektiğini araştırır.
Zamanla oyunun farklı toplumlarda farklı versiyonları türetilmiştir.
Anafartalar Zaferi takip etti.
Binlerce yeni okul inşa edildi.
.........
```

Figure 3.5 Wikipedia Turkish Sentence Dataset Sample

Quora dataset was also translated into Turkish. Since it only contains questions, the Wikipedia sentence dataset that was given in Figure 3.5, was used in order to create negative samples. At the end, two datasets were merged and labeled as questions or not, as shown in Figure 3.6.



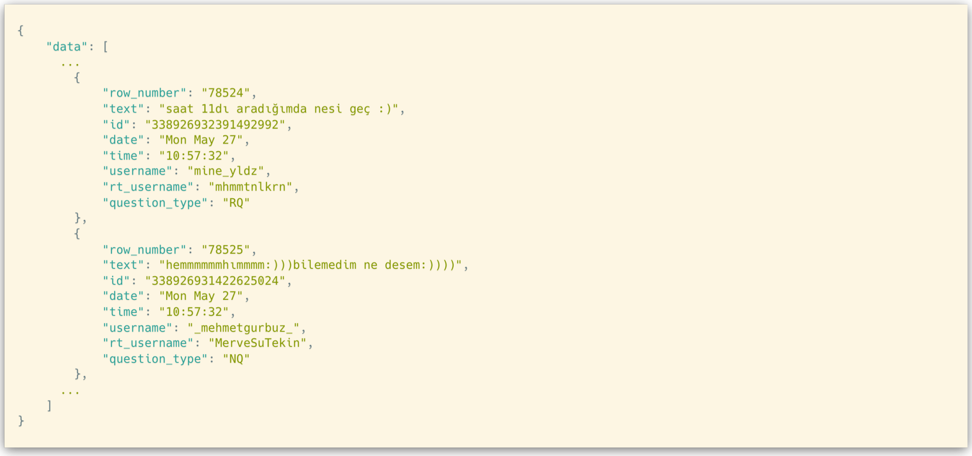
```
{
    "data": [
        {
            "text": "O sırada bir ayağını öbür ayağının üstüne koymuştu",
            "label": 0
        },
        {
            "text": "Küçük boyutlu bir İHA'yı şehrin sokaklarında uçurmak için hangi lisansa ihtiyacım var",
            "label": 1
        },
        {
            "text": "Masallarda olaylar bilinmeyen bir zamanda geçer",
            "label": 0
        },
        {
            "text": "Yeni çalışanların Farmington Bank'taki ilk günlerine girerken bilmeleri gereken bazı şeyler nelerdir",
            "label": 1
        },
        {
            "text": "Ardından Gaziantepspor`a transfer oldu",
            "label": 0
        },
        {
            "text": "İçeri giren hava dışarı çıkmak zorundadır",
            label": 0
        },
       ...
    ]
}
```

Figure 3.6 Quora-Wikipedia Mixed Dataset Sample

Özger et al. [53] created a huge tweet question dataset which tweeted within a month in 2013. They also categorized them by their meanings. The dataset structure can be found below:

- **row_number**: Row number of observations
- **text**: Tweet content
- **id**: Tweet id
- **date**: Tweet sent date
- **time**: Tweet sent time
- **username**: Username who tweet the content
- **rt_username**: Username who retweet original tweet
- **question_type**: Category of question

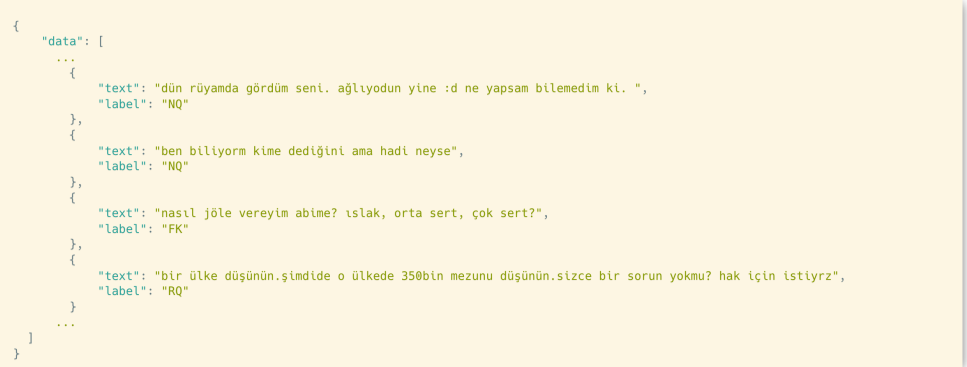
They separated questions into eight categories. Rhetorical questions (*RQ*) which OP doesn't want an answer, factual knowledge (*FK*) which OP seeks an answer, invitations (*IA*), recommendations (*RN*), questions with an answer (*QA*), opinions (*OP*), requests (*RT*), and no questions (*NQ*). Since some of these categories (IA, RN, QA, OP and RT) have relatively small sample sizes, we merged them as other questions (*OQ*). Some samples can be found in Figure 3.7.



```
{
    "data": [
      ...
        {
            "row_number": "78524",
            "text": "saat 11dı aradığımda nesi geç :)",
            "id": "338926932391492992",
            "date": "Mon May 27",
            "time": "10:57:32",
            "username": "mine_yldz",
            "rt_username": "mhmmtnlkrn",
            "question_type": "RQ"
        },
        {
            "row_number": "78525",
            "text": "hemmmmmmhımmmm:)))bilemedim ne desem:))))",
            "id": "338926931422625024",
            "date": "Mon May 27",
            "time": "10:57:32",
            "username": "_mehmetgurbuz_",
            "rt_username": "MerveSuTekin",
            "question_type": "NQ"
        },
      ...
    ]
}
```

Figure 3.7 Raw Tweet Question Dataset Sample

Tweet dataset was converted into JSON and unnecessary columns were deleted. Text data and corresponding labels are put together in a file as shown in the Figure 3.8



```
{
    "data": [
      ...
        {
            "text": "dün rüyamda gördüm seni. ağlıyodun yine :d ne yapsam bilemedim ki. ",
            "label": "NQ"
        },
        {
            "text": "ben biliyorm kime dediğini ama hadi neyse",
            "label": "NQ"
        },
        {
            "text": "nasıl jöle vereyim abime? ıslak, orta sert, çok sert?",
            "label": "FK"
        },
        {
            "text": "bir ülke düşünün.şimdide o ülkede 350bin mezunu düşünün.sizce bir sorun yokmu? hak için istiyrz",
            "label": "RQ"
        }
      ...
  ]
}
```

Figure 3.8 Tweet Question Dataset Sample

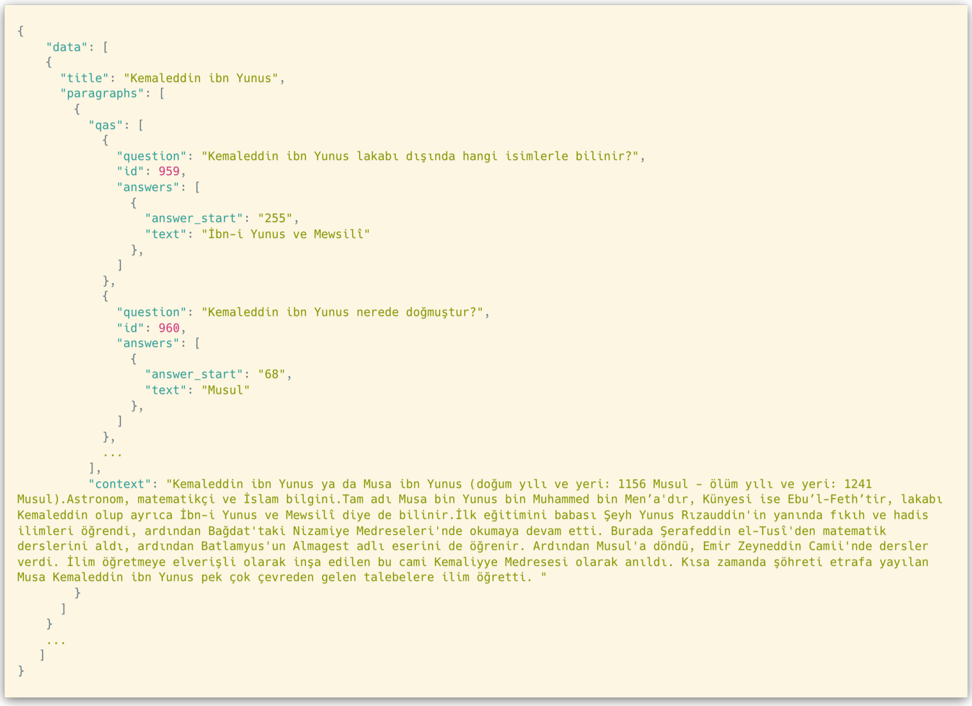
Finally, all three datasets are processed and separated into train, test, and validation sets for fine tuning with split sizes of 70%, 20%, and 10% respectively. The Table 3.1 summarizes the number of observations.

Dataset	**Category**	**Train Size**	**Validation Size**	**Test Size**	**Total**
QD-Dialog	Q	6.802	1.992	1.000	9.794
	NQ	19.092	5.406	2.700	27.198
QD-Quora	Q	6.486	1.772	903	9.161
	NQ	5.527	1.660	814	8.001
QD-Tweet	RQ	16.506	4.663	2.377	23.546
	FK	6.597	1.925	899	9.421
	OQ	8.436	2.350	1.190	11.976
	NQ	22.293	6.443	3.225	31.961

Table 3.1 Distribution of Dataset Instances for Text Classification

Question Answering

We obtained two question-answering datasets. Both of them are similar to SQuAD dataset [62], which is frequently used in question-answering systems. It was also shared as an open source [9]. The second dataset, which is a Wikipedia-based Turkish dataset, was provided by Prof. Diri [23]. These two sources were used to fine-tune the question-answering task.



```
{
    "data": [
    {
      "title": "Kemaleddin ibn Yunus",
      "paragraphs": [
        {
          "qas": [
            {
              "question": "Kemaleddin ibn Yunus lakabı dışında hangi isimlerle bilinir?",
              "id": 959,
              "answers": [
                {
                  "answer_start": "255",
                  "text": "İbn-i Yunus ve Mewsilî"
                },
              ]
            },
            {
              "question": "Kemaleddin ibn Yunus nerede doğmuştur?",
              "id": 960,
              "answers": [
                {
                  "answer_start": "68",
                  "text": "Musul"
                },
              ]
            },
            ...
          ],
          "context": "Kemaleddin ibn Yunus ya da Musa ibn Yunus (doğum yılı ve yeri: 1156 Musul - ölüm yılı ve yeri: 1241
Musul).Astronom, matematikçi ve İslam bilgini.Tam adı Musa bin Yunus bin Muhammed bin Men’a'dır, Künyesi ise Ebu’l-Feth’tir, lakabı
Kemaleddin olup ayrıca İbn-i Yunus ve Mewsilî diye de bilinir.İlk eğitimini babası Şeyh Yunus Rızauddin'in yanında fıkıh ve hadis
ilimleri öğrendi, ardından Bağdat'taki Nizamiye Medreseleri'nde okumaya devam etti. Burada Şerafeddin el-Tusî'den matematik
derslerini aldı, ardından Batlamyus'un Almagest adlı eserini de öğrenir. Ardından Musul'a döndü, Emir Zeyneddin Camii'nde dersler
verdi. İlim öğretmeye elverişli olarak inşa edilen bu cami Kemaliyye Medresesi olarak anıldı. Kısa zamanda şöhreti etrafa yayılan
Musa Kemaleddin ibn Yunus pek çok çevreden gelen talebelere ilim öğretti. "
        }
      ]
    }
    ...
   ]
}
```

Figure 3.9 Question Answering Dataset Structure

The dataset contains list of paragraphs/contexts and questions that are related to them. A sample was given in Figure 3.9. Each question could have one or more answers, and *answer_start* property indicates the starting index of the answer in the context. The Table 3.2 summarizes the number of observations for question-answering datasets.

Dataset	Type	Train Size	Validation Size	Test Size	Total
QA-TQuad	Context	7.688	2.197	1.099	10.984
	Questions	482.460	136.129	67.250	685.839
QA-YTU	Context	3.517	1.005	503	5.025
	Questions	181.182	52.400	25.566	259.148

Table 3.2 Distribution of Dataset Instances for Question Answering

Feature Extraction

Visual analysis is a widely used method to provide information about the content of data. First of all, the features of the data to be visualized should be extracted and shaped according to the chart types to be used. The character counts, word counts and part-of-speech (PoS) tags for text classification datasets, and the character and word counts of answers for question answer datasets were obtained. Afterwards, histograms and scatter plots were drawn with the help of the `matplotlib` library in Python.

PoS tagging is a technique used in linguistics to determine the type of a word. The PoS tags of each sentence were extracted and recorded with the library named Zemberek [5] developed for Turkish. During this process, it is expected that the sentences would be written grammatically correct so that the word types can be properly separated. Since the tweet dataset contains irregular sentences, tokenization was applied before obtaining PoS tags.

Since the datasets we collected are relatively large, the PoS tagging process has been accelerated by using the API service written on Zemberek [11]. Using the `/find_pos` method of the this service, results as in Figure 3.10 were obtained.

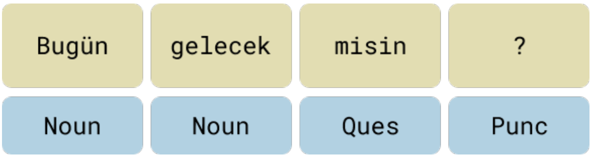


Figure 3.10 PoS Tagging

As can be seen in Figure 3.11 the library could not determine the adjacent word ”gelecekmisin”.

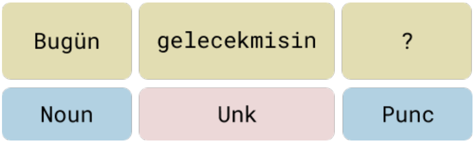


Figure 3.11 PoS Tagging without Tokenization

After tokenization, PoS tags were successfully obtained and it can be seen in Figure 3.12.

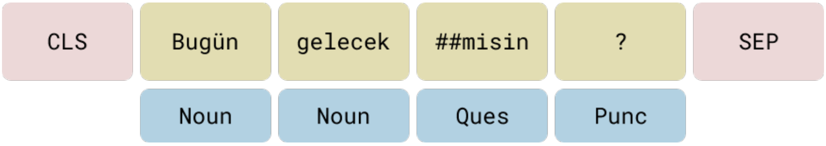


Figure 3.12 PoS Tagging with Tokenization

3.2.2. Fine-Tuning

Training the entire model requires excessive computation power. Thus, pre-trained models has been proposed. These models are trained with large corpora. For instance, approximately 4.5 billion tokens were used to train the BERTurk model [72]. We used cased versions of four different models from this study which are Bert, DistilBert, ConvBert, Electra respectively. Since Turkish is accent sensitive, cased version of these models was selected.

Using output weights as initial layer weights in a neural network and adjusting hyper-parameters is called fine-tuning. Huggingface's transformers library provides many useful functionalities and makes the fine-tuning process easy. The process starts with preparing the dataset by generating encodings with tokenizer. After that, models are trained with given hyper-parameters, and finally, predictions are made with fine-tuned models.

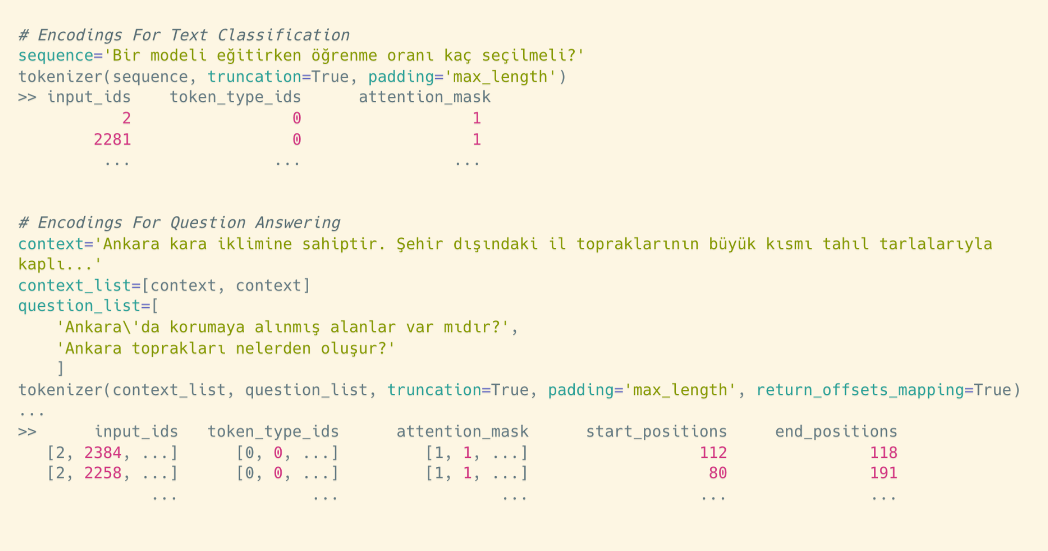
Data Preparation

The dataset preparation process for training differs from NLP task to task. It is sufficient to simply pass the sentence that is classified to the tokenizer for the sentence classification task. This process splits the given sentence into tokens, and as a result, three different pieces of information are obtained, which are `input_id`, `token_type_id` and `attention_mask` respectively.

In question answering datasets, the context and questions about the context are passed to the tokenizer as a list. In addition to sentence classification, `start_positions` and `end_positions` information, which shows at what position the answers starts and ends in the context, are also included in the dataset to be trained. (see [36]). The description of these input parameters can be found as follows:

- **input_ids:** Numerical representation of sequence tokens.
- **token_type_ids:** Indicator used to separate sequences in models that take two different sequences. e.g. `[CLS] SEQUENCE_A [SEP] SEQUENCE_B`.
- **attention_mask:** An index to tell the model which token it should use.
- **start_positions:** Starting index of the answer in the context.
- **end_positions:** End index of the answer in the context.

When `truncation` option is set `True` provided in the tokenizer, sentences that exceed the model's maximum token limit are truncated. `padding='max_length'` option fills the sentence's token vector with a special `[PAD]` token in order to adjust the encoding vector to maximum size of the model. `return_offset_mapping` is required for the question answering model and provides indexes that indicate where each token begins and ends. The overall process and results were given in Figure 3.13.



```
# Encodings For Text Classification
sequence='Bir modeli eğitirken öğrenme oranı kaç seçilmeli?'
tokenizer(sequence, truncation=True, padding='max_length')
>> input_ids    token_type_ids     attention_mask
          2                  0                  1
       2281                  0                  1
        ...                ...                ...


# Encodings For Question Answering
context='Ankara kara iklimine sahiptir. Şehir dışındaki il topraklarının büyük kısmı tahıl tarlalarıyla
kaplı...'
context_list=[context, context]
question_list=[
    'Ankara\'da korumaya alınmış alanlar var mıdır?',
    'Ankara toprakları nelerden oluşur?'
    ]
tokenizer(context_list, question_list, truncation=True, padding='max_length', return_offsets_mapping=True)
...
>>       input_ids   token_type_ids      attention_mask       start_positions      end_positions
   [2, 2384, ...]      [0, 0, ...]         [1, 1, ...]                   112                118
   [2, 2258, ...]      [0, 0, ...]         [1, 1, ...]                    80                191
              ...              ...                 ...                   ...                ...
```

Figure 3.13 Python Code Snippet for Extracting Encodings

Training and Evaluation

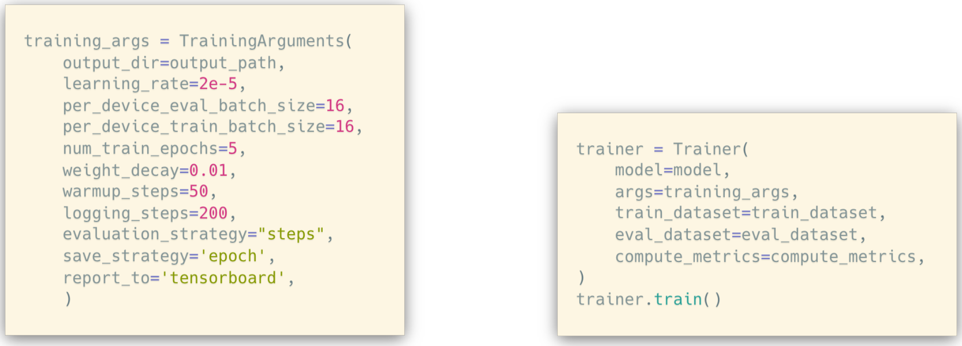
Text classification and question answering models was fine-tuned with the help of `Trainer` class provided by Huggingface's library. Google Colab Pro, which has an NVIDIA Tesla P100 GPU processor, was used with the following setup to train these models:

Task	**Epochs**	**Batch Size**	**Learning Rate**
Text Classification	5	16	2e-5
Question Answering	5	16	3e-5

Table 3.3 Hyper-parameter Setup for NLP Tasks

The parameters required for the training of the sentence classification task were determined as in Figure 3.14. Learning rate is a hyper-parameter that controls how frequently the weights are updated throughout the training phase. Epoch is the number of repetitions of training and the batch size is number of samples to be fed to the learning algorithm.

The model was trained by creating the `Trainer` class with the `compute_metric` function, along with the training arguments, training and validation sets. The hyper-parameters given in Table 3.3 were determined in the `TrainingArguments` class and it was adjusted to calculate the metrics at each step and to save the model at each epoch. In addition, the setting that will enable all metrics to be displayed on the tensorboard is determined by the `report_to` parameter.



```
training_args = TrainingArguments(
    output_dir=output_path,
    learning_rate=2e-5,
    per_device_eval_batch_size=16,
    per_device_train_batch_size=16,
    num_train_epochs=5,
    weight_decay=0.01,
    warmup_steps=50,
    logging_steps=200,
    evaluation_strategy="steps",
    save_strategy='epoch',
    report_to='tensorboard',
    )
```

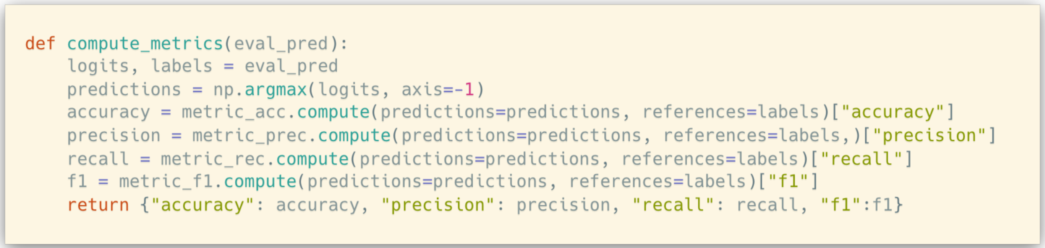
(a) Training Arguments

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    compute_metrics=compute_metrics,
)
trainer.train()
```

(b) Trainer

Figure 3.14 Python Code Snippet for Trainer Class in Text Classification

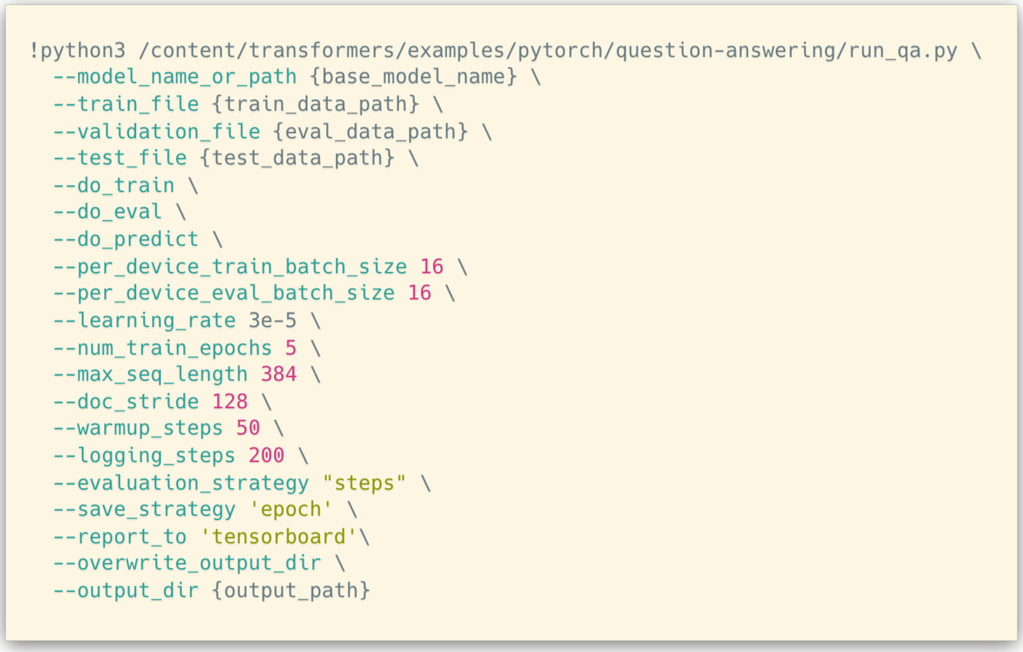
The four metrics determined for text classification and they were calculated with `compute_metrics` function as can be seen in Figure 3.15.



```python
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    accuracy = metric_acc.compute(predictions=predictions, references=labels)["accuracy"]
    precision = metric_prec.compute(predictions=predictions, references=labels,)["precision"]
    recall = metric_rec.compute(predictions=predictions, references=labels)["recall"]
    f1 = metric_f1.compute(predictions=predictions, references=labels)["f1"]
    return {"accuracy": accuracy, "precision": precision, "recall": recall, "f1":f1}
```

Figure 3.15 Python Code Snippet for Compute Metric Function

Training codes, which were created by Huggingface, were used in order to train question-answering datasets. The training was done with the hyper-parameters in Table 3.3 from the command line, as can be seen in Figure 3.16. The Exact Match and F1 metrics are calculated for question answering and the results are given in Section 4.



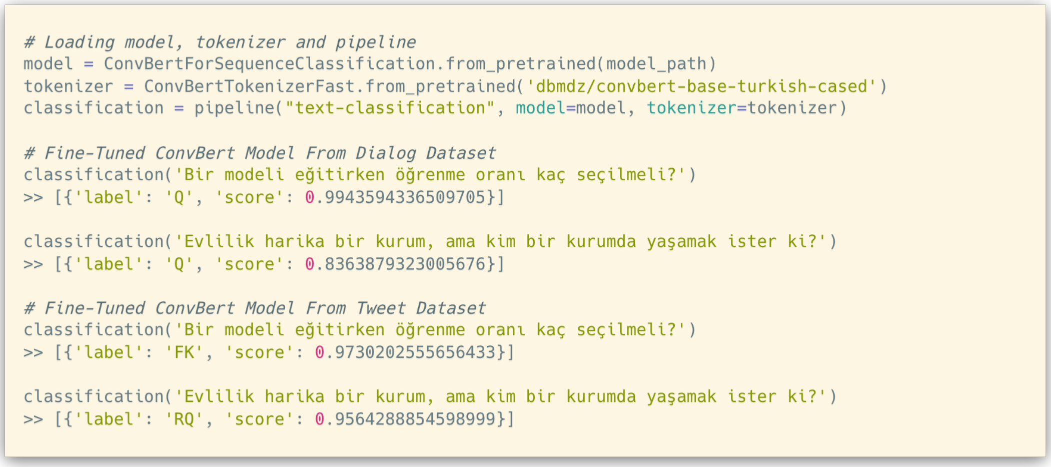
```
!python3 /content/transformers/examples/pytorch/question-answering/run_qa.py \
  --model_name_or_path {base_model_name} \
  --train_file {train_data_path} \
  --validation_file {eval_data_path} \
  --test_file {test_data_path} \
  --do_train \
  --do_eval \
  --do_predict \
  --per_device_train_batch_size 16 \
  --per_device_eval_batch_size 16 \
  --learning_rate 3e-5 \
  --num_train_epochs 5 \
  --max_seq_length 384 \
  --doc_stride 128 \
  --warmup_steps 50 \
  --logging_steps 200 \
  --evaluation_strategy "steps" \
  --save_strategy 'epoch' \
  --report_to 'tensorboard'\
  --overwrite_output_dir \
  --output_dir {output_path}
```

Figure 3.16 Python Code Snippet for Trainer Class in Question Answering

Inference

`pipeline` function provided by the Huggingface's transformers library were used for inference. This function basically takes model, tokenizer, sequence and task type as input and then it makes prediction and calculates probabilities with softmax function.

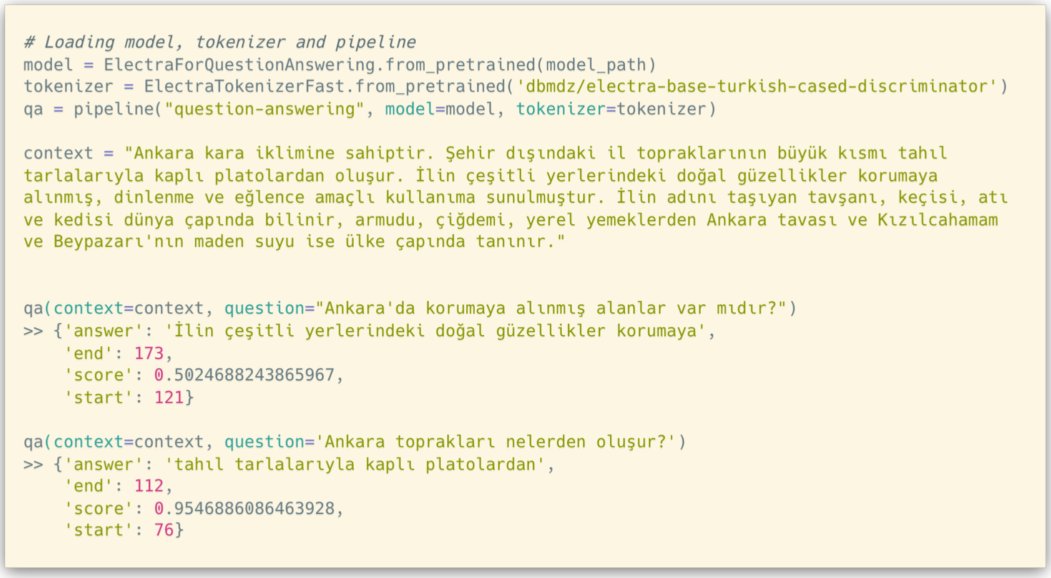
Three different data sets were used to fine-tune the text classification model. QD-Dialog and QD-Quora are datasets prepared for binary classification, while QD-Tweet is a multi-class classification dataset. As can be seen in Figure 3.17, binary classification datasets also positively predicted rhetorical questions, the type of question for which a person does not seek an answer.



```
# Loading model, tokenizer and pipeline
model = ConvBertForSequenceClassification.from_pretrained(model_path)
tokenizer = ConvBertTokenizerFast.from_pretrained('dbmdz/convbert-base-turkish-cased')
classification = pipeline("text-classification", model=model, tokenizer=tokenizer)

# Fine-Tuned ConvBert Model From Dialog Dataset
classification('Bir modeli eğitirken öğrenme oranı kaç seçilmeli?')
>> [{'label': 'Q', 'score': 0.9943594336509705}]

classification('Evlilik harika bir kurum, ama kim bir kurumda yaşamak ister ki?')
>> [{'label': 'Q', 'score': 0.8363879323005676}]

# Fine-Tuned ConvBert Model From Tweet Dataset
classification('Bir modeli eğitirken öğrenme oranı kaç seçilmeli?')
>> [{'label': 'FK', 'score': 0.9730202555656433}]

classification('Evlilik harika bir kurum, ama kim bir kurumda yaşamak ister ki?')
>> [{'label': 'RQ', 'score': 0.9564288854598999}]
```

Figure 3.17 Python Code Snippet for Inference in Text Classification

Question answering model takes two inputs as context and question. In Figure 3.18, a paragraph taken from Wikipedia was defined as the context and two questions were asked and the answers were obtained with the help of the fine-tuned model. The predicted response from the `pipeline` result, the starting and ending indexes of the response and the prediction score can be obtained.

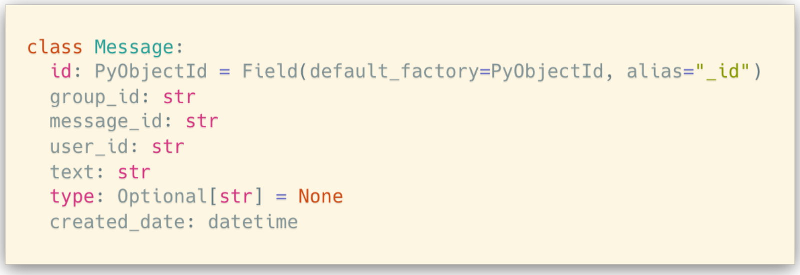


```
# Loading model, tokenizer and pipeline
model = ElectraForQuestionAnswering.from_pretrained(model_path)
tokenizer = ElectraTokenizerFast.from_pretrained('dbmdz/electra-base-turkish-cased-discriminator')
qa = pipeline("question-answering", model=model, tokenizer=tokenizer)

context = "Ankara kara iklimine sahiptir. Şehir dışındaki il topraklarının büyük kısmı tahıl
tarlalarıyla kaplı platolardan oluşur. İlin çeşitli yerlerindeki doğal güzellikler korumaya
alınmış, dinlenme ve eğlence amaçlı kullanıma sunulmuştur. İlin adını taşıyan tavşanı, keçisi, atı
ve kedisi dünya çapında bilinir, armudu, çiğdemi, yerel yemeklerden Ankara tavası ve Kızılcahamam
ve Beypazarı'nın maden suyu ise ülke çapında tanınır."


qa(context=context, question="Ankara'da korumaya alınmış alanlar var mıdır?")
>> {'answer': 'İlin çeşitli yerlerindeki doğal güzellikler korumaya',
    'end': 173,
    'score': 0.5024688243865967,
    'start': 121}

qa(context=context, question='Ankara toprakları nelerden oluşur?')
>> {'answer': 'tahıl tarlalarıyla kaplı platolardan',
    'end': 112,
    'score': 0.9546886086463928,
    'start': 76}
```

Figure 3.18 Python Code Snippet for Inference in Question Answering

3.3. Message Pre-Processing

Before starting to process the messages in any social media applications, it is necessary to store them in a database. The main reason for this is that social media applications provide on-demand access with the help of APIs instead of giving access to all messages. Therefore, `conversation_history` method that is provided by the Slack API was used to collect chat messages and then they were stored in MongoDB with a structure that can be found in Figure 3.19.



```python
class Message:
  id: PyObjectId = Field(default_factory=PyObjectId, alias="_id")
  group_id: str
  message_id: str
  user_id: str
  text: str
  type: Optional[str] = None
  created_date: datetime
```

Figure 3.19 Python Code Snippet of Message Class

While messages being saved into the database, a cron job checks the system, finds messages after the last processed message, and processes them in batches of 200. Messages were processed with NLP tasks that is given in Section 2.5. The text classification model was used in the first step to determine whether each message is a question; in the second step, the message subject was associated with the sender using zero-shot classification. In the last step, the answers to the previously found questions were searched by using a question-answering model. This processing flow was described visually in the Figure 3.20.

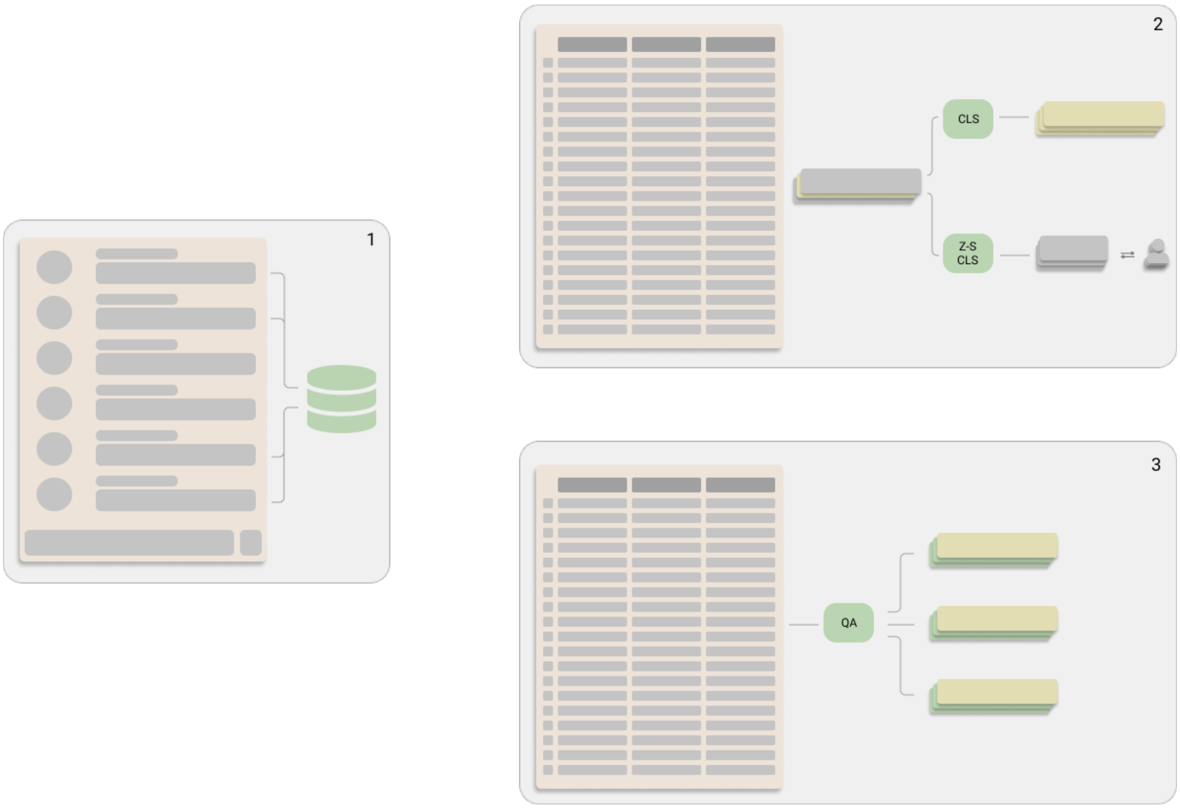


Figure 3.20 Message Processing Flow

Similar questions from the user can be searched for through the vector search feature in Elasticsearch. This feature works by calculating the distance between the vectors consisting of numerical data obtained from the sentences. Semantic properties should be preserved while extracting numerical data. Reimers et al. claimed that the BERT structure is unsuitable for this process and proposed the Siamese BERT approach [63]. With this technique, vectors can be obtained without losing the semantic properties of the sentence. The 384 dimensional dense vectors were obtained for each question, and they were recorded in Elasticsearch. The `sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2` model which can be found in Huggingface was used to extract these vectors.

The questions in chat groups are usually answered right after they have been asked. However, they could be mixed up with the other messages during regular conversation on a certain subject. We proposed an algorithm to overcome this difficulty. By taking into consideration the token limit of the BERT model [77], which is 512, we created a candidate answer pool from messages that are sent after the detected questions. Then, lists containing 2, 5, and 10 sentences were created from the candidate answer pool. The question-answering model was used on the contexts created from the sentences in these lists, and the answers with the highest scores were saved.

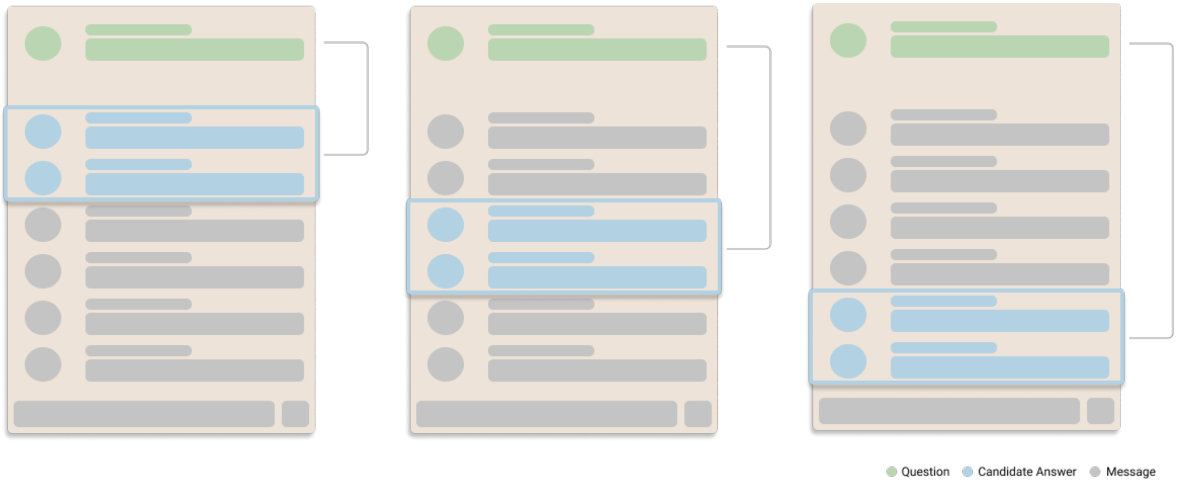
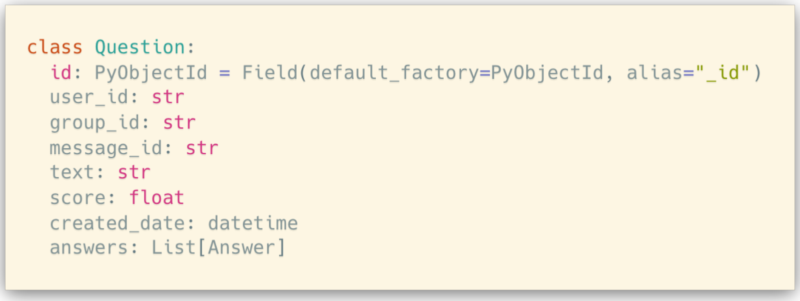


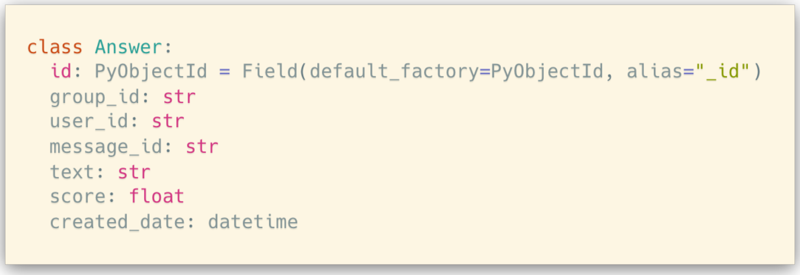
Figure 3.21 Chat Group Conversation Windows

Figure 3.22 and 3.23 show the final question and answer data model used within application.



```python
class Question:
  id: PyObjectId = Field(default_factory=PyObjectId, alias="_id")
  user_id: str
  group_id: str
  message_id: str
  text: str
  score: float
  created_date: datetime
  answers: List[Answer]
```

Figure 3.22 Python Code Snippet of Question Class

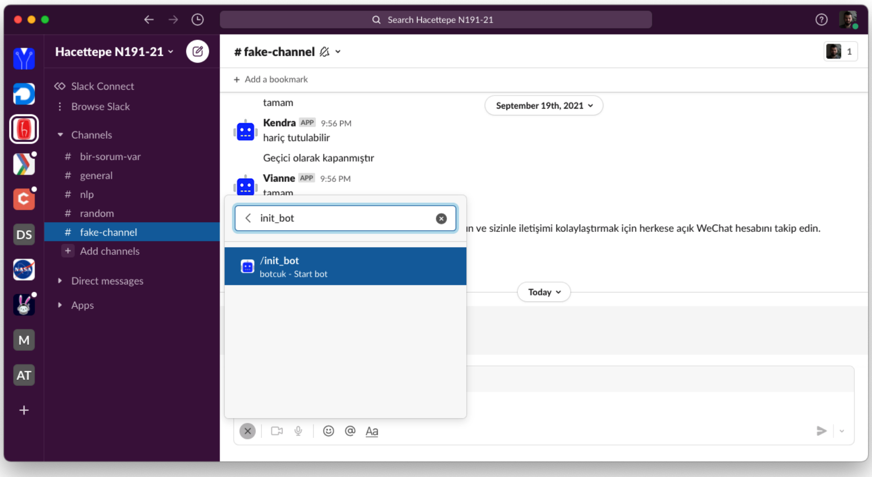


```python
class Answer:
  id: PyObjectId = Field(default_factory=PyObjectId, alias="_id")
  group_id: str
  user_id: str
  message_id: str
  text: str
  score: float
  created_date: datetime
```

Figure 3.23 Python Code Snippet of Answer Class

3.4. Production

The application module was built on Slack. There are two types of functionalities. The first one, called `/init_bot` that is responsible for setting up the channel or group, and it can only be executed by admins. The other one `/ask` is for regular users. It takes a text input that should be a question and initiates a similar question-finding process.



tamam

Kendra APP 9:56 PM
hariç tutulabilir

Geçici olarak kapanmıştır

Vianne APP 9:56 PM
tamam

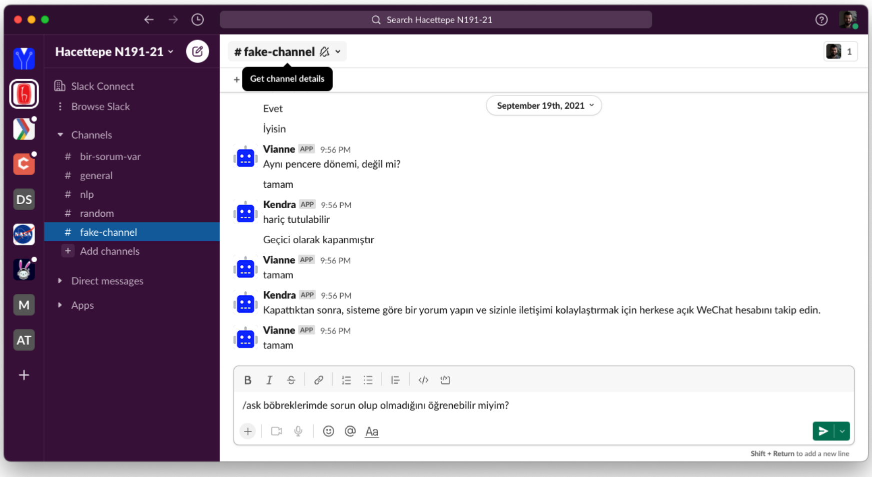
init_bot

/init_bot
botcuk - Start bot

ın ve sizinle iletişimi kolaylaştırmak için herkese açık WeChat hesabını takip edin.

Figure 3.24 Initializing Bot

As can be seen in Figure 3.24, after executing `init_bot` function, a combo-box appears and the admin selects any group to start pre-processes, which were described in Section 3.3.



Evet

İyisin

Vianne APP 9:56 PM
Aynı pencere dönemi, değil mi?

tamam

Kendra APP 9:56 PM
hariç tutulabilir

Geçici olarak kapanmıştır

Vianne APP 9:56 PM
tamam

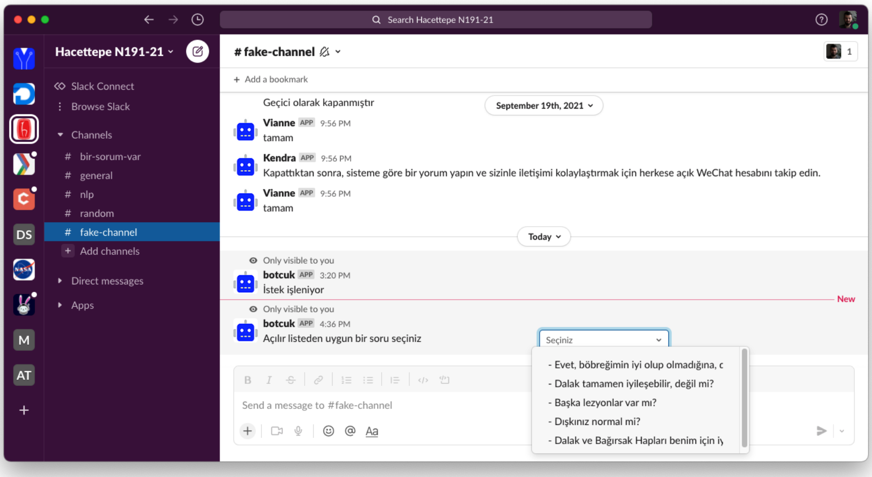
Kendra APP 9:56 PM
Kapattıktan sonra, sisteme göre bir yorum yapın ve sizinle iletişimi kolaylaştırmak için herkese açık WeChat hesabını takip edin.

Vianne APP 9:56 PM
tamam

/ask böbreklerimde sorun olup olmadığını öğrenebilir miyim?

Figure 3.25 Asking Question

The usage of `/ask` function was given in Figure 3.25. Once a user asks a question by using `/ask` function, the application displays in a combo-box the 5 most similar questions that are similarity score ordered, as can be seen in Figure 3.26.



Geçici olarak kapanmıştır

Vianne APP 9:56 PM
tamam

Kendra APP 9:56 PM
Kapattıktan sonra, sisteme göre bir yorum yapın ve sizinle iletişimi kolaylaştırmak için herkese açık WeChat hesabını takip edin.

Vianne APP 9:56 PM
tamam

Only visible to you
botcuk APP 3:20 PM
İstek işleniyor

Only visible to you
botcuk APP 4:36 PM
Açılır listeden uygun bir soru seçiniz

Seçiniz

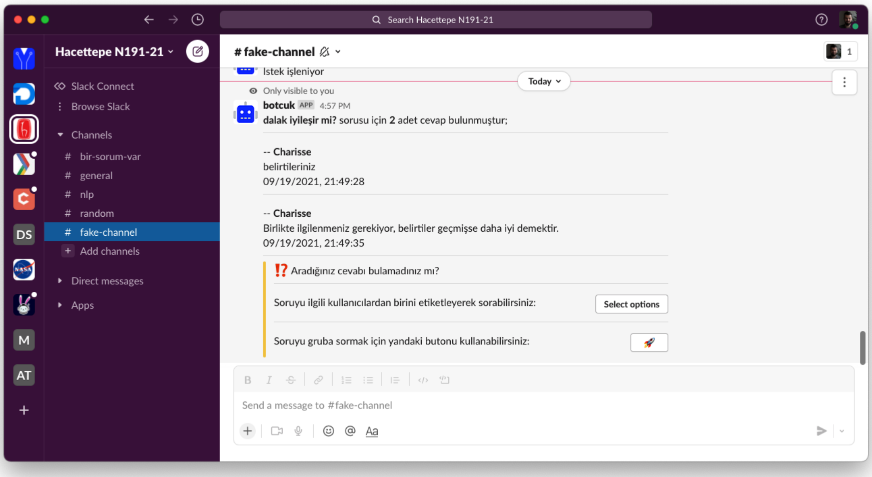
- Evet, böbreğimin iyi olup olmadığına, c
- Dalak tamamen iyileşebilir, değil mi?
- Başka lezyonlar var mı?
- Dışkınız normal mi?
- Dalak ve Bağırsak Hapları benim için iy

Figure 3.26 Similar Question Selection

Similar questions were found by vector search of Elasticsearch. It uses the cosine function to measure the similarity of two vectors. Cosine similarity can be calculated as follows:

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}} \tag{16}$$

where $\mathbf{A}$ and $\mathbf{B}$ represent input vectors. The closer the result is to 1, the two vectors are said to be similar.



İstek işleniyor

Only visible to you

botcuk APP 4:57 PM

dalak iyileşir mi? sorusu için **2** adet cevap bulunmuştur;

-- **Charisse**
belirtileriniz
09/19/2021, 21:49:28

-- **Charisse**
Birlikte ilgilenmeniz gerekiyor, belirtiler geçmişse daha iyi demektir.
09/19/2021, 21:49:35

⁉️ Aradığınız cevabı bulamadınız mı?

Soruyu ilgili kullanıcılardan birini etiketleyerek sorabilirsiniz:

Soruyu gruba sormak için yandaki butonu kullanabilirsiniz:

Figure 3.27 Displaying Answers

Finally, answers to the selected question were displayed to the user as in Figure 3.27. All of the results were only shown to the user without disturbing the group. In the last stage, if the desired answer was not found, the user can send the question to the group by tagging the person most relevant to the topic or send it directly without tagging.

4. Results

The only requirement for DL models to work properly is labeled data. Collecting and creating labeled data is a very expensive and laborious process. In this study, the models specified in Section 2.4. were fine-tuned by both translating the datasets prepared in other languages into Turkish and including the previously prepared datasets.

4.1. Text Classification

For the training of the text classification task, the data should be separated into certain classes and labeled. Since the focus of our study is to understand whether a sentence is a question or not, three different datasets were prepared within the scope of this focus. These prepared data were fine tuned with Google Colab Pro service and their codes were shared as open source [46]. Fine-tuning process was done with a batch size of size 16, and a learning rate of 2e-5 for 5 epochs, which all these are selected through the preliminary experimentation.

4.1.1. Dialog Dataset

Exploratory Analysis

It can be seen in the Figure 4.1 that the average number of characters in the dialog dataset, which has a total of 36,992 samples of data, is 40. It has been noted that in the the data labeled as Question has longer character size.

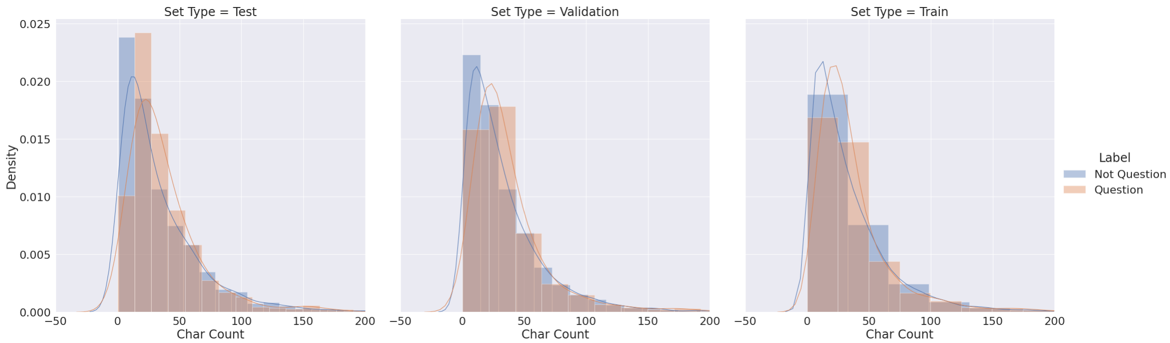


Figure 4.1 Dailog Dataset Character Count Distribution

As can be seen in Figure 4.2 that the average word count is 5. In the test and validation data, the sentences labeled as non-question with shorter sizes are denser, while there is a more balanced distribution in the training data. All sets have a right skewed distribution.

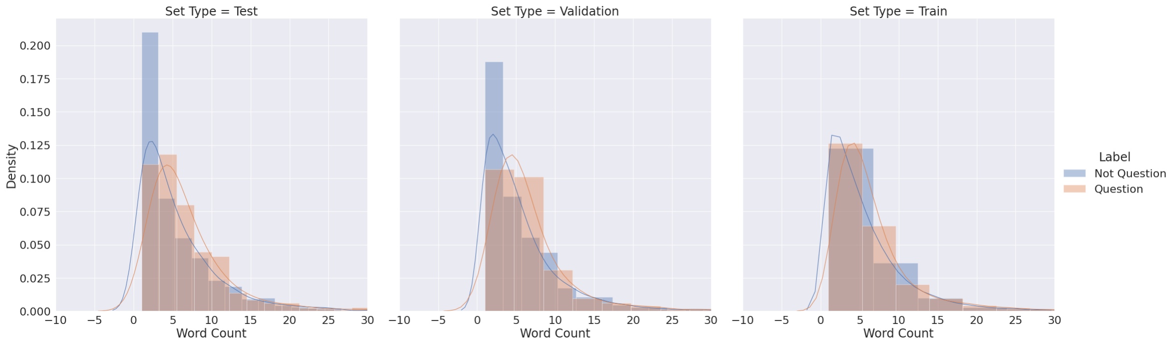


Figure 4.2 Dailog Dataset Word Count Distribution

In Figure 4.3, which was created by extracting sentence structure, it was observed that verbs and adjectives were dense in the sentences labeled as not question in the training data. On the other hand, the number of nouns in the questions is higher.

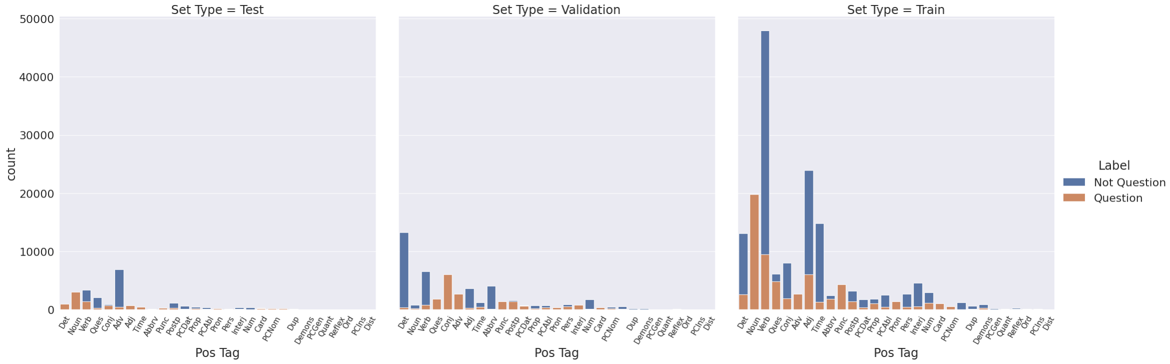
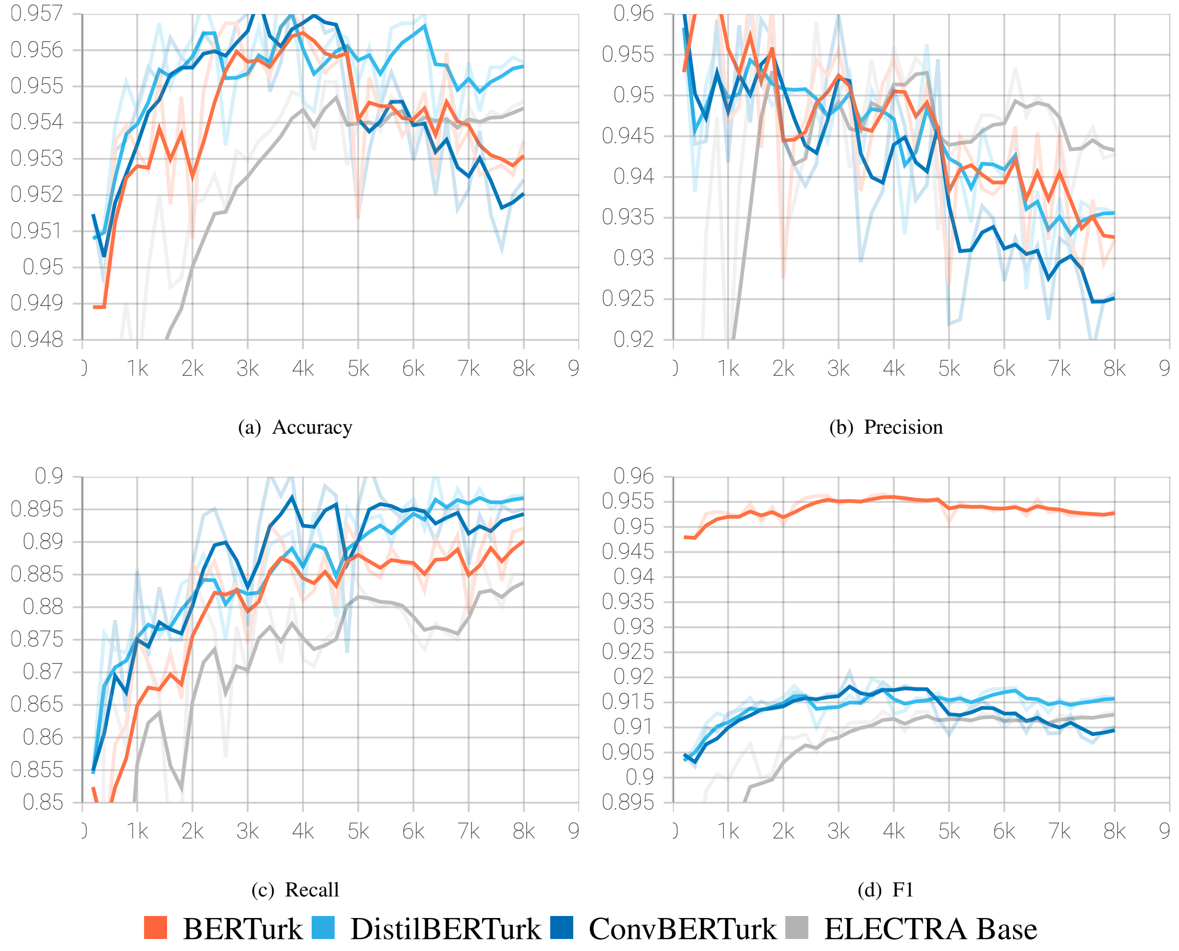


Figure 4.3 Dailog Dataset Pos Tags

Fine Tuning

The fine-tuning process, in which four different models were used, was completed in eight thousand steps. It was observed that in the Figure 4.4, the accuracy decreases after the 4,000th step (3rd epoch). In the F1 metric, while the base model has high values, other models produced lower values, and they are similar to each other.



(a) Accuracy

(b) Precision

(c) Recall

(d) F1

Figure 4.4 Dialog Dataset Fine Tune Metrics by Step Charts

In the last stage, the BERTTurk base model obtained the highest F1 score on the training data. The model that achieved the largest F1 score on the test data was ConvBERTurk. The model that achieved the highest accuracy was again the ConvBERTurk model. It was seen that in Table 4.1, the accuracy values are very close to each other.

Model	Phase	Accuracy	Precision	Recall	F1
BERTurk	Train/Evaluation	0.957015	0.951456	0.885542	**0.956515**
	Test	0.961081	0.950317	0.902610	0.925849
DistilBERTurk	Train/Evaluation	0.957556	0.947705	0.891566	0.918779
	Test	0.962432	0.952481	0.905622	0.928461
ConvBERTurk	Train/Evaluation	**0.958773**	0.951311	0.892570	0.921005
	Test	**0.963243**	0.956475	0.904618	**0.929824**
ELECTRA Base	Train/Evaluation	0.955123	0.953057	0.876506	0.913180
	Test	0.958378	0.942226	0.900602	0.920944

Table 4.1 Dialog Dataset Fine Tuning Metrics

4.1.2. Quora Dataset

Exploratory Analysis

Quora dataset has 17,162 samples. Figure 4.5 shows that all sets are balanced and has an average of 50 characters on each sequence.

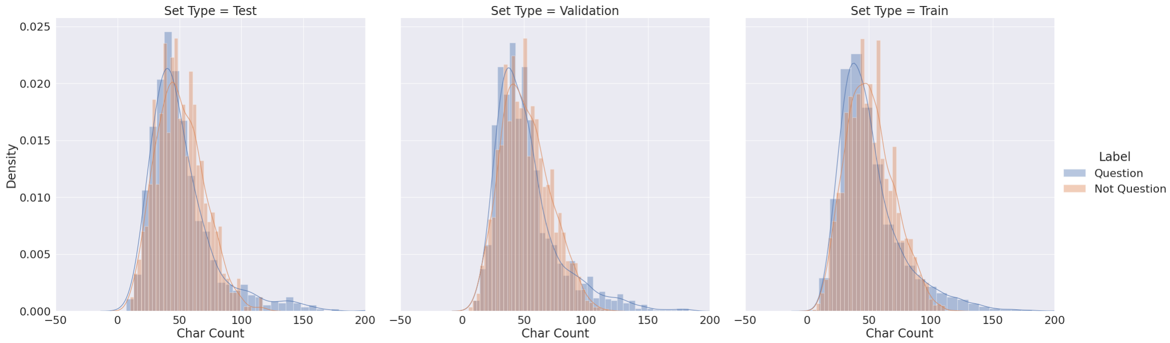


Figure 4.5 Quora Dataset Character Count Distribution

If we look at word distribution in Figure 4.6, the sequences that are labeled as ”not in question” have 9 to 10 words on average. The ”question” sequences have 5 to 6 words on average. Again, the distribution seems balanced.

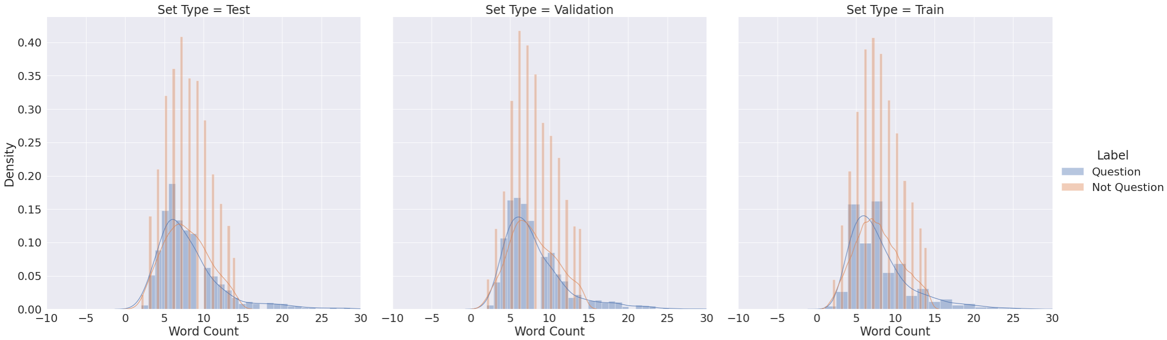


Figure 4.6 Quora Dataset Word Count Distribution

The distribution of sentence structure demonstrates that ”question” sequences generally include adverbs, but ”not question” sentences mostly contain nouns which can be seen in Figure 4.7.

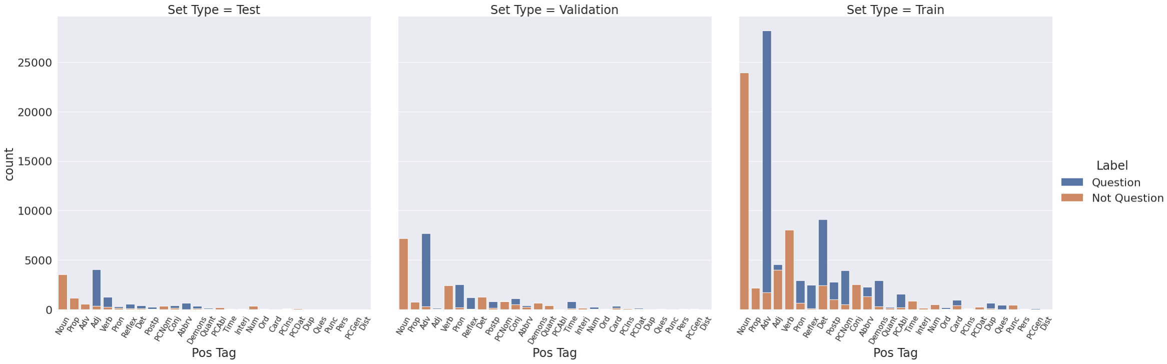
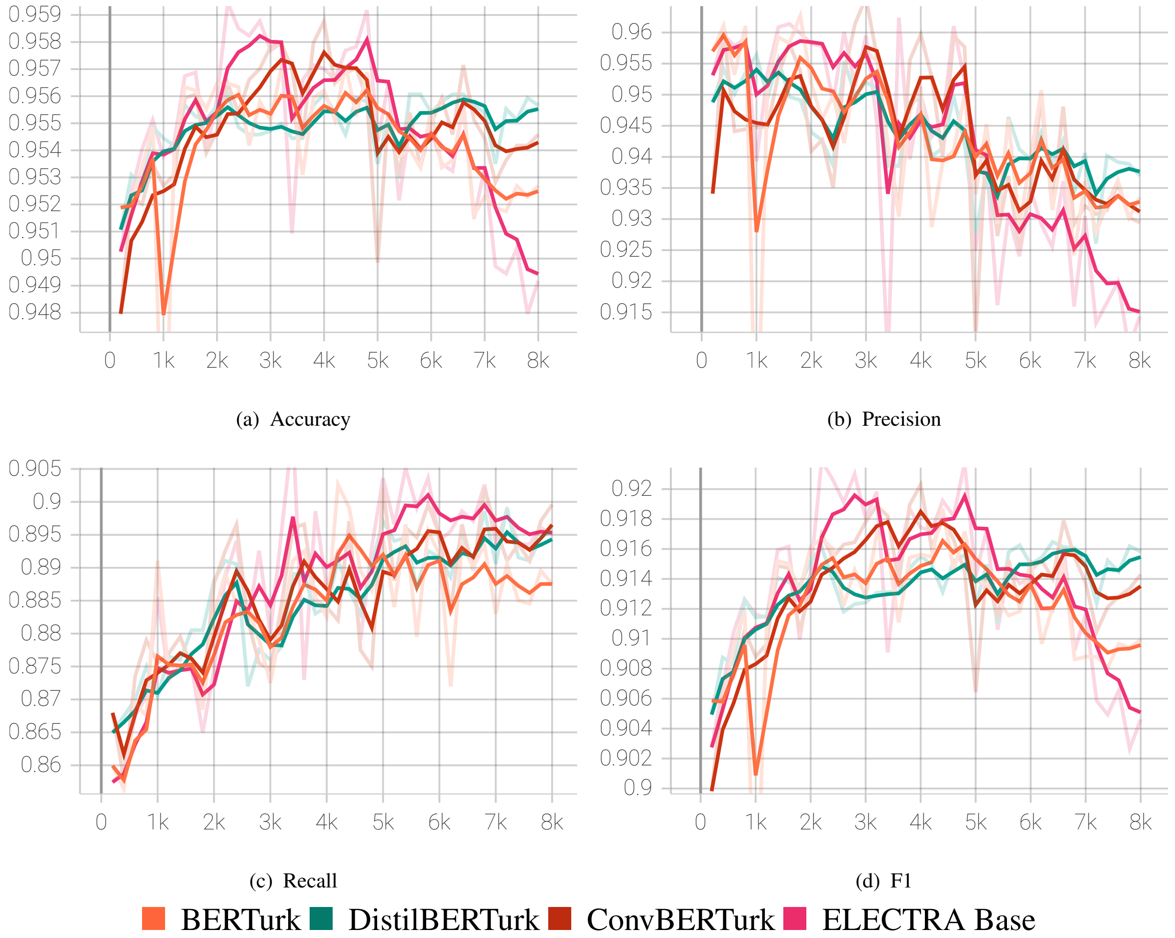


Figure 4.7 Quora Dataset Pos Tags

Fine Tuning

According to Figure 4.8, the process was completed in eight thousand steps for each model. It is obvious that accuracy and F1 metrics started to decrease after the third epoch. The performance of the four models is similar to each other.



(a) Accuracy

(b) Precision

(c) Recall

(d) F1

Figure 4.8 Quora Dataset Fine Tune Metrics by Step Charts

The ELECTRA Base model achieved the greatest Accuracy and F1 score on the training data in the final stage. BERTurk was the model with the highest Accuracy and F1 score on the test data. The final results can be found in Table 4.2.

Model	Phase	Accuracy	Precision	Recall	F1
BERTurk	Train/Evaluation	0.957151	0.939171	0.899096	0.918697
	Test	**0.994758**	0.998883	0.991140	**0.994997**
DistilBERTurk	Train/Evaluation	0.956100	0.956400	0.899100	0.916100
	Test	0.980198	0.997709	0.964562	0.980855
ConvBERTurk	Train/Evaluation	0.958773	0.959673	0.884036	0.920303
	Test	0.983110	.997722	.970099	0.983717
ELECTRA Base	Train/Evaluation	**0.959178**	0.952355	0.893072	**0.921762**
	Test	0.980198	0.997709	0.964562	0.980855

Table 4.2 Quora Dataset Fine Tuning Metrics

4.1.3. Tweet Dataset

Exploratory Analysis

The hand-crafted Tweet dataset has 76,904 samples with four different classes: ”not a question”, ”rhetoric question”, ”other question” and ”factual knowledge”. The character count distribution which was given in Figure 4.9, shows that the maximum character length is 150 and the average is 40 to 50.

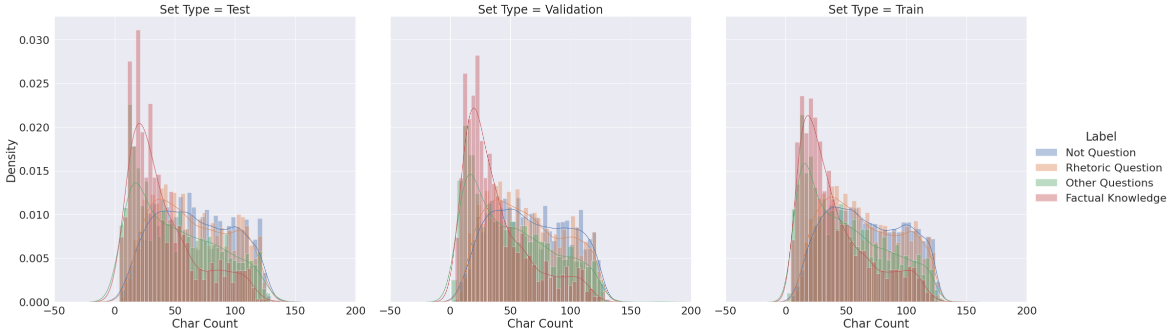


Figure 4.9 Tweet Dataset Character Count Distribution

Figure 4.10 shows that the average word count is 10. Test, train and validation sets have similar distributions. The average word count of the sequences with the label ”factual knowledge” is 5, which is a good indicator that the real questions are made up of short sentences.

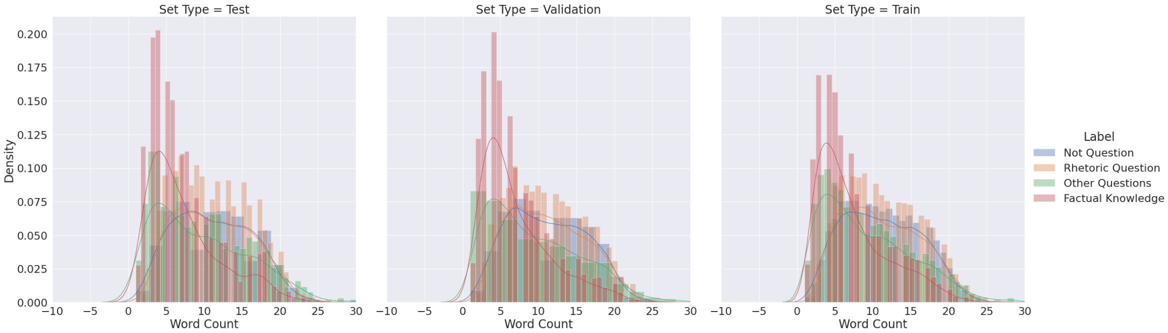


Figure 4.10 Tweet Dataset Word Count Distribution

As in the Figure 4.11, the Pos Tags distribution of Tweet dataset shows that ”not question” sequences mostly made up of abbreviations. The reason for this could be that the tweet data consists of random sentences.

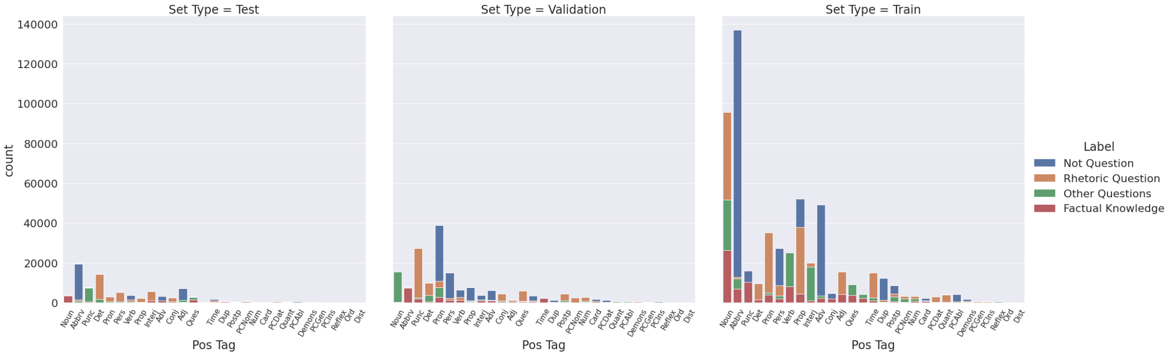
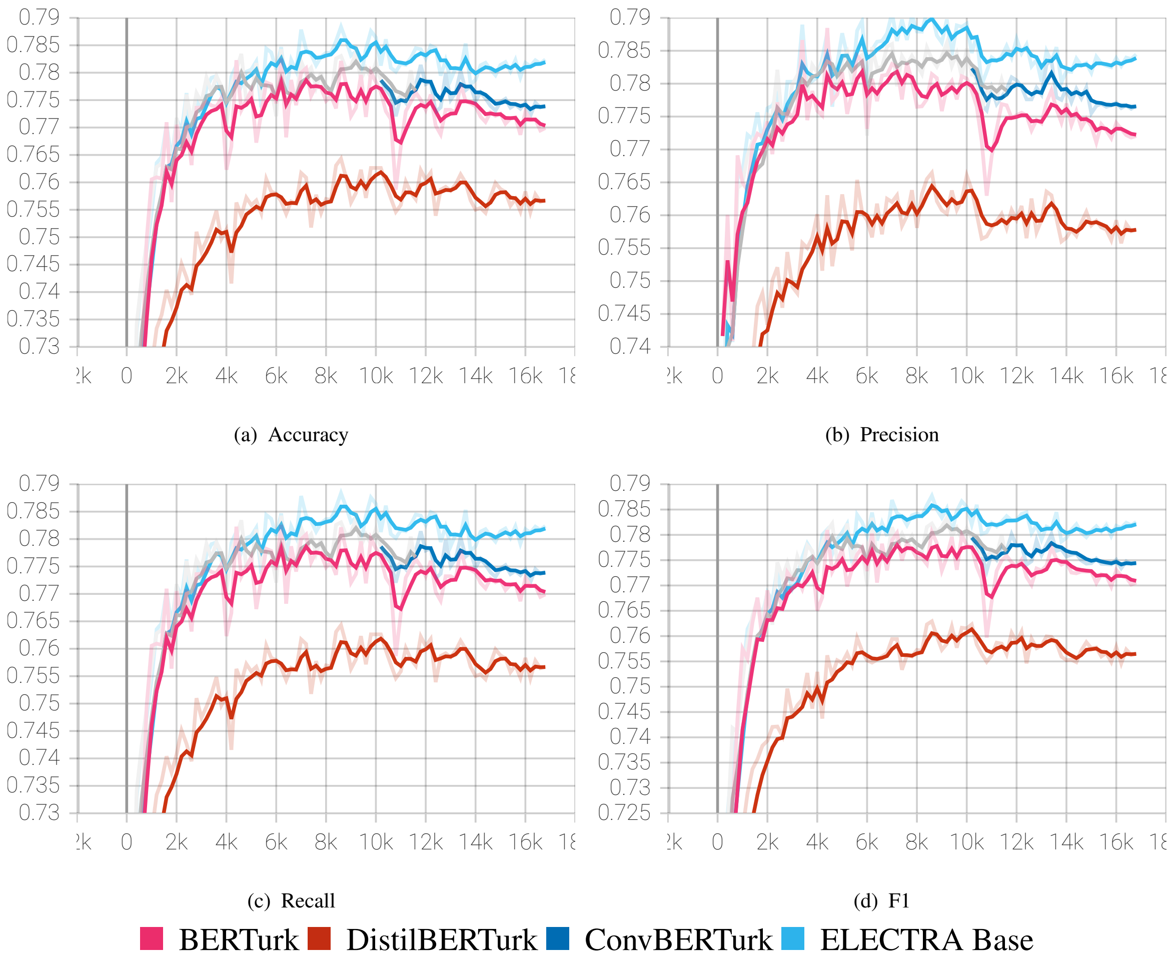


Figure 4.11 Tweet Dataset Pos Tags

Fine Tuning

For each model, fine-tuning took sixteen thousand steps. DistilBERTurk shows bad performance on this dataset. The performance of the other models is similar, but the ELECTRA Base is superior to the others. The fine-tuning metrics by steps can be found in Figure 4.12.



(a) Accuracy

(b) Precision

(c) Recall

(d) F1

Figure 4.12 Tweet Dataset Fine Tune Metrics by Step Charts

In the final step, the ELECTRA Base model had the highest Accuracy and F1 score on the training data. On the test data, ConvBERTurk had the greatest Accuracy and F1 score. The results were given in Table 4.3.

Model	Phase	Accuracy	Precision	Recall	F1
BERTurk	Train/Evaluation	0.782134	0.781457	0.782134	0.780599
	Test	0.781042	0.782017	0.781042	0.781219
DistilBERTurk	Train/Evaluation	0.764300	0.766500	0.764300	0.762900
	Test	0.760239	0.761716	0.760239	0.760213
ConvBERTurk	Train/Evaluation	0.783800	0.786000	0.782000	0.781900
	Test	**0.782082**	0.784239	0.782082	**0.782424**
ELECTRA Base	Train/Evaluation	**0.788375**	0.790655	0.788375	**0.787725**
	Test	0.781302	0.783859	0.781302	0.781766

Table 4.3 Tweet Dataset Fine Tuning Metrics

4.2. Question Answering

Question answering is an NLP task that provides answers to questions about a paragraph by searching within that paragraph. Training data consists of paragraphs, possible questions that can be asked to the paragraph, and start and end indexes of answers. Fine-tuning was performed with the help of the sample fine-tuning algorithm shared by Huggingface with a batch size of size 16, and a learning rate of 3e-5 for 5 epochs [35].

4.2.1. TQuad Dataset

Exploratory Analysis

The average number of characters in the TQuad dataset, which comprises a total of 10,984 paragraph and 68,5839 question, is 30 to 40, as shown in Figure 4.13.

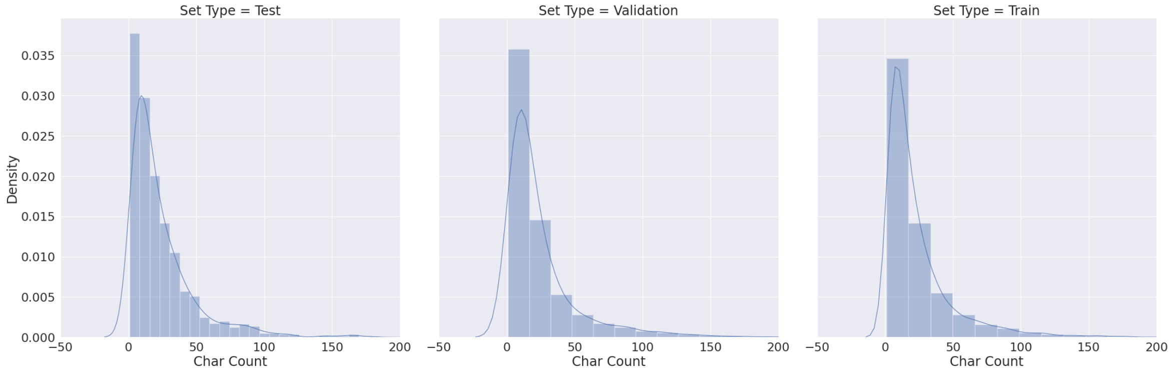


Figure 4.13 TQuad Dataset Character Count Distribution of Answers

Figure 4.14 shows that the average word count is 3 to 4. Distributions of all three set are right skewed.

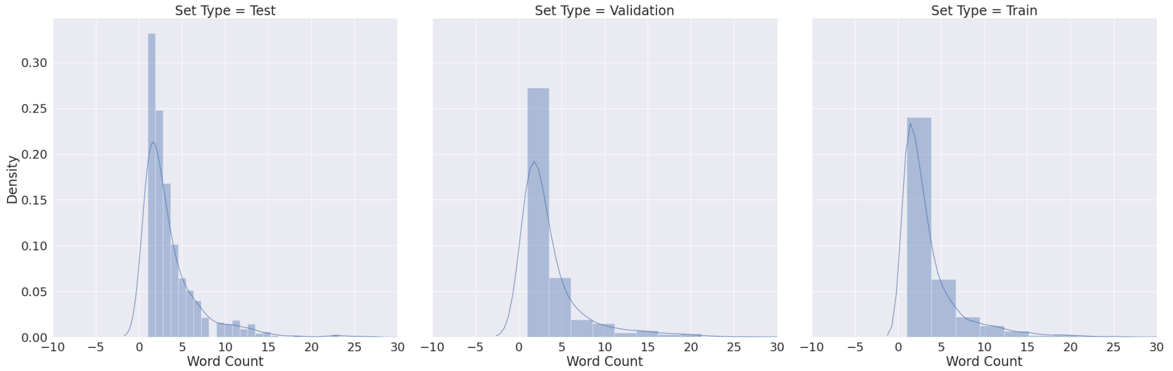
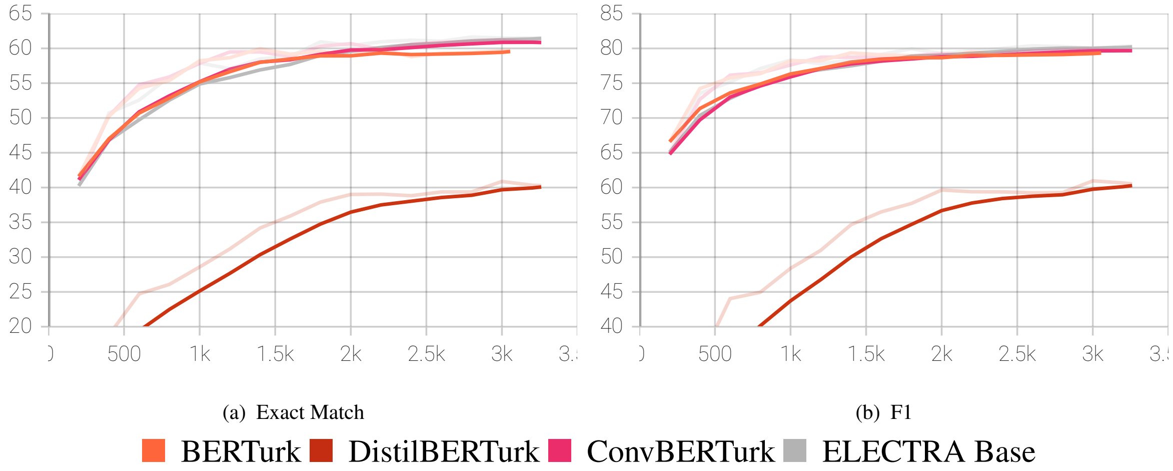


Figure 4.14 TQuad Dataset Word Count Distribution of Answers

Fine Tuning

Fine-tuning process for each model is completed in three thousand steps which can been in 4.15. On this dataset, DistilBERTurk performs poorly. The performance of the other models is comparable, but the ELECTRA Base outperforms them all.



(a) Exact Match

(b) F1

Figure 4.15 TQuad Dataset Fine Tune Metrics by Step Charts

Table 4.4 shows the ELECTRA Base model outperforms all other models.

Model	**Phase**	**Exact Match**	**F1**
BERTurk	Train/Evaluation	59.7178	79.4480
	Test	59.4177	79.6924
DistilBERTurk	Train/Evaluation	40.2822	60.5264
	Test	39.7634	59.0327
ConvBERTurk	Train/Evaluation	60.8102	79.6843
	Test	60.6005	79.3676
ELECTRA Base	Train/Evaluation	**61.5385**	**80.3351**
	Test	**61.2375**	**80.7814**

Table 4.4 TQuad Dataset Fine Tuning Metrics

4.2.2. YTU QA Dataset

Exploratory Analysis

YTU dataset consists of 5,025 paragraph and 259,148 question. Figure 4.16 shows the average character count of 20 to 30, and Figure 4.17 shows the average word count of 3 to 4.

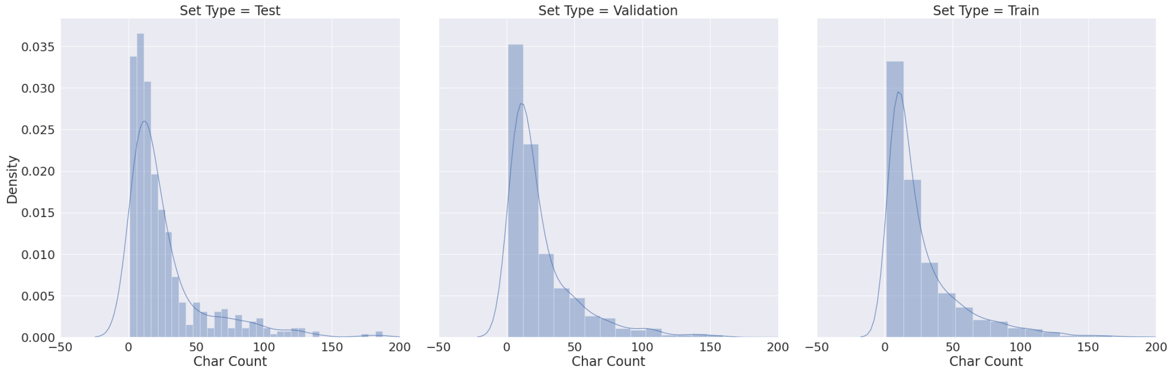


Figure 4.16 YTU QA Dataset Character Count Distribution of Answers

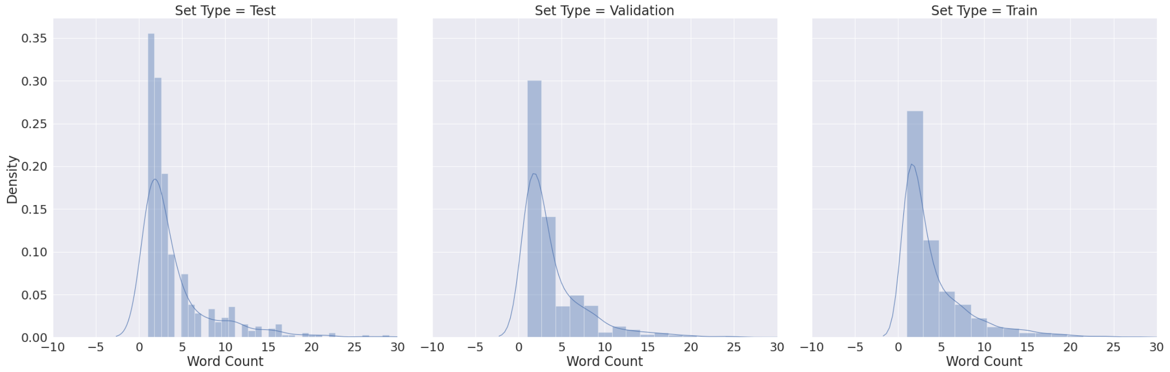
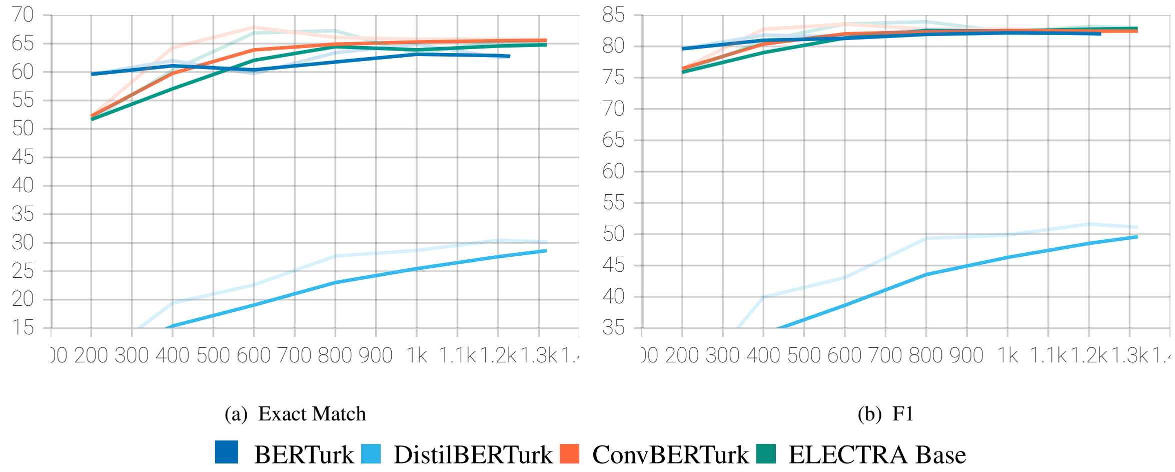


Figure 4.17 YTU QA Dataset Word Count Distribution of Answers

Fine Tuning

Each model's fine-tuning procedure takes one thousand three hundred steps. Again, DistilBERTurk is a model with the lowest performance. The performance of the other models is comparable. The performance metric chart was given in Figure 4.18.



(a) Exact Match

(b) F1

Figure 4.18 YTU QA Dataset Fine Tune Metrics by Step Charts

Final results in the Table 4.5, show that ELECTRA Base has the highest score on the train dataset. On the other hand, ConvBERTurk model shows the best performance on the test dataset.

Model	**Phase**	**Exact Match**	**F1**
BERTurk	Train/Evaluation	62.5871	81.8977
	Test	56.2624	76.7202
DistilBERTurk	Train/Evaluation	30.1493	51.1165
	Test	29.8211	48.2728
ConvBERTurk	Train/Evaluation	65.6716	82.4270
	Test	**62.0278**	**79.8566**
ELECTRA Base	Train/Evaluation	**65.0746**	**82.9919**
	Test	60.0398	77.9323

Table 4.5 YTU QA Dataset Fine Tuning Metrics

5. Conclusion

Communication, one of the basic needs of today, has been moved to social media platforms with the digital revolution. When these platforms, which are used as communication tools, and people's searches for information are combined, a large pool of textual data emerges. NLP is used as a technology that can explain the meaning and summarize the complicated information in this pool. Natural language models using deep learning methods as their basis succeed in natural language tasks such as sentence classification, question answering, named entity recognition, paragraph summarization, etc. The biggest reason for this success is the discovery of pre-trained models such as BERT. These models are created by training for days with large-volume datasets, namely corpora, prepared in the target language, and can be used in downstream tasks by fine-tuning.

Fine-tuning is possible with customized datasets for downstream tasks. Since our study related to sentence classification and question answering tasks, 5 different Turkish datasets have been prepared, 3 of which are used in sentence classification and 2 of which are used in question answering tasks. Preparing a labeled dataset is a laborious task, and it is not easy to find labeled data for every language. For this reason, the Dialog dataset used in the study was translated from Chinese and the Quora dataset was translated into Turkish from English with the help of the Google Translate API. Fine-tuning was done in the Colab environment by using the prepared datasets and a pre-trained BERTurk model for Turkish.

As a result of fine tuning, ConvBERT model trained with Dialog dataset in sentence classification task has the highest Accuracy. But it is assumed that this model will also classify rhetorical questions as real questions. Therefore, it is considered more appropriate to use the ELECTRA Base model trained with the Tweet dataset for this task. For the question answering task, the ELECTRA Base model trained using the YTU dataset showed the highest performance.

In the light of all these developments, an application was created to find similar questions and answers in a group and to display them to the user. Slack was used as a social media platform due to its superior features. The application, which allows user interaction, consists of two main parts. The first part analyzes the messages with NLP methods and saves them in the database. During the analysis process, the questions are separated using sentence classification, the answers to the questions separated using the question-answering task, and the interests of the users are categorized with zero-shot classification. The second piece of the application acts as a bridge that communicates with the Slack API. By changing only this part, the application can be used on different social media platforms.

The performance of the resulting application is directly proportional to the performance of the models used. In other words, the better the performance of the models used for the NLP tasks mentioned above, the better the experience can be offered to the users. The biggest factor that increases the performance of the models is the preparation of labeled and clean data. It was seen that adapting it to Turkish by following the studies in other languages is important in terms of closing the labeled data gap in the literature.

REFERENCES

[1] Ai-powered automation for service and support teams. **2021**. Accessed April 30, 2022. https://www.talla.com/.

[2] Question answering on squad1.1, **2021**. Accessed May 14, 2022. https://paperswithcode.com/sota/question-answering-on-squad11.

[3] An open source conversational ai framework. **2022**. Accessed April 30, 2022. https://deeppavlov.ai/.

[4] Aksoy, A. Turkish sentences for word2vec training, **2016**. Accessed March 23, 2022. https://www.kaggle.com/datasets/ahmetax/hury-dataset.

[5] Akın, A. A. ahmetaa/zemberek-nlp: Nlp tools for turkish. **2021**. Accessed April 10, 2022. https://github.com/ahmetaa/zemberek-nlp.

[6] Alammar, J. The illustrated transformer, **2018**. Accessed April 28, 2022. http://jalammar.github.io/illustrated-transformer/.

[7] Alammar, J. A visual guide to using bert for the first time, **2019**. Accessed April 29, 2022. https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/.

[8] Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. **2016**. doi:10.48550/ARXIV.1607.06450. https://arxiv.org/abs/1607.06450.

[9] Balki, C. Turkish nlp squad repo. **2021**. Accessed January 29, 2022. https://github.com/cbalkig/Turkish-NLP-SQuAD-Repo.

[10] Bellman, R. *Dynamic Programming*. Dover Publications, Inc, **2003**.

[11] Bilgili, C. cbilgili/zemberek-nlp-server: Zemberek türkçe nlp java kütüphanesi üzerine rest docker sunucu, **2018**. Accessed April 10, 2022. https://github.com/cbilgili/zemberek-nlp-server.

[12] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, **2020**. doi:10.48550/ARXIV.2005.14165. https://arxiv.org/abs/2005.14165.

[13] Canuma, P. The brief history of nlp - datadriveninvestor. **2019**. Accessed March 02, 2022. https://medium.datadriveninvestor.com/the-brief-history-of-nlp-c90f331b6ad7.

[14] Caswell, I. and Liang, B. Recent advances in google translate. **2020**. Accessed February 13, 2022. https://ai.googleblog.com/2020/06/recent-advances-in-google-translate.html.

[15] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv.org*, **2014**.

[16] Christensen, H., Gotoh, Y., and Renals, S. Punctuation annotation using statistical prosody models. *University of Sheffield*, **2001**.

[17] Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*. **2020**. https://openreview.net/forum?id=r1xMH1BtvB.

[18] Collobert, R. and Weston, J. A unified architecture for natural language processing. *In Proceedings of the 25th International Conference on Machine Learning*, pages 160–167, **2008**.

[19] Commons, W. File:recurrent neural network unfold.svg — wikimedia commons, the free media repository. **2021**. Accessed April 23,

2022. https://commons.wikimedia.org/w/index.php?title=File:Recurrent_neural_network_unfold.svg&oldid=605590767.

[20] Da, P. What is the difference between machine comprehension and question answering in nlp? **2019**. Accessed January 29, 2022. https://www.quora.com/What-is-the-difference-between-machine-comprehension-and-question-answering-in-NLP.

[21] Deng, L. and Liu, Y. *Deep Learning in Natural Language Processing*. Springer Publishing Company, Incorporated, 1st. ed. edition, **2018**.

[22] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv.org*, **2018**. https://arxiv.org/abs/1810.04805.

[23] Diri, B. Wikipedia based question answering dataset. **2022**. Accessed March 26, 2022. https://avesis.yildiz.edu.tr/iletisim.

[24] dontloo (https://stats.stackexchange.com/users/95569/dontloo). What exactly are keys, queries, and values in attention mechanisms? Cross Validated, **2019**. https://stats.stackexchange.com/q/424127. URL:https://stats.stackexchange.com/q/424127 (version: 2021-12-31).

[25] EmSa1998. Internet forums are basically dead and that's a big loss. **2016**. Accessed March 7, 2022. https://www.reddit.com/r/SeriousConversation/comments/fbvivk/internet_forums_are_basically_dead_and_thats_a/.

[26] Fischer, T. and Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, **2018**. doi:10.1016/j.ejor.2017.11.054. https://www.sciencedirect.com/science/article/abs/pii/S0377221717310652.

[27] Geron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Online Learning, 2nd. ed. edition, **2019**. https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.

[28] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, **2016**. http://www.deeplearningbook.org.

[29] Google. Dialogflow documentation, **2022**. Accessed April 30, 2022. https://cloud.google.com/dialogflow/docs.

[30] Google. Translating text (basic). **2022**. Accessed February 13, 2022. https://cloud.google.com/translate/docs/basic/quickstart#translate_translate_text-drest.

[31] He, P., Gao, J., and Chen, W. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. **2021**. doi:10.48550/ARXIV.2111.09543. https://arxiv.org/abs/2111.09543.

[32] Horev, R. Bert explained: State of the art language model for nlp. Towardsdatascience.com. **2018**. Accessed January 29, 2022. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[33] Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. *arXiv.org*, **2018**. https://arxiv.org/abs/1801.06146.

[34] Hsieh, T.-J., Hsiao, H.-F., and Yeh, W.-C. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied Soft Computing*, 11(2):2510–2525, **2011**. doi:10.1016/j.asoc.2010.09.007.

[35] huggingface. Question answering, **2022**. Accessed January 29, 2022. https://github.com/huggingface/transformers/tree/main/examples/pytorch/question-answering.

[36] huggingface. Question answering example, **2022**. Accessed April 17, 2022. https://github.com/huggingface/transformers/tree/main/examples/pytorch/question-answering.

[37] Jia, Q. Qa matching. Github.com. **2020**. Accessed January 29, 2022. https://github.com/JiaQiSJTU/QAmatching.

[38] Jia, Q., Zhang, M., Zhang, S., and Zhu, K. Q. Matching questions and answers in dialogues from online forums. *arXiv.org*, **2020**. doi:10.3233/FAIA200326. https://arxiv.org/abs/2005.09276.

[39] Jiang, P., Muppalla, K. S., Wei, Q., Gopal, C. N., and Wang, C. Double-barreled question detection at momentive. *arXiv.org*, **2022**. doi:10.48550/arXiv.2203.03545. https://arxiv.org/abs/2203.03545.

[40] Jiang, Z., Yu, W., Zhou, D., Chen, Y., Feng, J., and Yan, S. Convbert: Improving bert with span-based dynamic convolution, **2020**. doi:10.48550/ARXIV.2008.02496. https://arxiv.org/abs/2008.02496.

[41] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., and et. al. In-datacenter performance analysis of a tensor processing unit. *arXiv.org*, **2017**. https://arxiv.org/abs/1704.04760.

[42] Kleczek, D. Question answering tutorial. Kaggle.com. **2021**. Accessed January 29, 2022. https://www.kaggle.com/thedrcat/question-answering-tutorial.

[43] Kraus, M., Feuerriegel, S., and Oztekin, A. Deep learning in business analytics and operations research: Models, applications and managerial implications. *European Journal of Operational Research*, 281(3):628–641, **2020**. doi:10.1016/j.ejor.2019.09.018.

[44] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., **2012**. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[45] Kwong, H. and Yorke-Smith, N. Detection of imperative and declarative question–answer pairs in email conversations. *AI Communications*, 25(4):271–283, **2012**. doi:10.3233/aic-2012-0516. https://content.iospress.com/articles/ai-communications/aic516.

[46] Kılıç, I. Berturk performance analysis on text classification and question answering tasks in turkish datasets. Github.com. **2022**. Accessed May 7, 2022. https://github.com/izzetkalic/botcuk-dataset-analyze.

[47] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, **2015**. doi:10.1038/nature14539. https://www.nature.com/articles/nature14539.

[48] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach, **2019**. doi:10.48550/ARXIV.1907.11692. https://arxiv.org/abs/1907.11692.

[49] Margolis, A. and Ostendorf, M. Question detection in spoken conversations using textual conversations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 118–124. Association for Computational Linguistics, Portland, Oregon, USA, **2011**. https://aclanthology.org/P11-2021.

[50] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space, **2013**. doi:10.48550/ARXIV.1301.3781. https://arxiv.org/abs/1301.3781.

[51] Munnangi, M. A comprehensive guide to nlp. - towards data science. Towardsdatascience.com. **2019**. Accessed January 29, 2022. https://towardsdatascience.com/nlp-with-spacy-part-1-beginner-guide-to-nlp-4b9460652994.

[52] Owens, J., Houston, M., Luebke, D., Green, S., Stone, J., and Phillips, J. Gpu computing. *Proceedings of the IEEE*, 96:879–899, **2008**. doi:10.1109/JPROC.2008.917757.

[53] Ozger, Z. B., Diri, B., and Girgin, C. Question identification on turkish tweets. In *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*, pages 126–130. **2014**. doi:10.1109/INISTA.2014.6873608.

[54] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. *arXiv.org*, **2018**. https://arxiv.org/abs/1802.05365.

[55] Phi, M. Illustrated guide to transformers- step by step explanation, **2020**. Accessed May 8, 2022. https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0.

[56] Prasoon, S. A simple introduction to sequence to sequence models. analyticsvidhya.com. **2020**. Accessed February 19, 2022. https://www.analyticsvidhya.com/blog/2020/08/a-simple-introduction-to-sequence-to-sequence-models/.

[57] Quora. Quora question pairs. **2017**. Accessed March 23, 2022. https://www.kaggle.com/competitions/quora-question-pairs/submit.

[58] Radford, A. Improving language understanding with unsupervised learning. **2018**. https://openai.com/blog/language-unsupervised/.

[59] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. *OpenAI*, **2018**. Accessed May 14, 2022. https://openai.com/blog/language-unsupervised/.

[60] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI*, **2019**. Accessed

May 14, 2022. https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[61] Raina, R., Madhavan, A., and Ng, A. Y. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 873–880. Association for Computing Machinery, New York, NY, USA, **2009**. ISBN 9781605585161. doi:10.1145/1553374.1553486. https://doi.org/10.1145/1553374.1553486.

[62] Rajpurka, P. The stanford question answering dataset (squad). **2021**. Accessed January 29, 2022. https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/.

[63] Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, **2019**. http://arxiv.org/abs/1908.10084.

[64] Rizvi, M. S. Z. Demystifying bert: A comprehensive guide to the groundbreaking nlp framework, **2019**. https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/.

[65] Rohrer, B. Transformers from scratch. **2021**. Accessed April 28, 2022. https://e2eml.school/transformers.html.

[66] Ruder, S. NLP's ImageNet moment has arrived. **2018**. Accessed January 29, 2022. https://ruder.io/nlp-imagenet/.

[67] Ruder, S. A review of the recent history of natural language processing, **2018**. https://ruder.io/a-review-of-the-recent-history-of-nlp/index.html.

[68] Ruder, S. QA—how did we get here? , adapting to time , data detectives., **2021**. Accessed January 29, 2022. https://newsletter.ruder.io/issues/qa-how-did-we-get-here-adapting-to-time-data-detectives-379447.

[69] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, **2019**. doi:10.48550/ARXIV.1910.01108. https://arxiv.org/abs/1910.01108.

[70] Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, **2015**. doi:10.1016/j.neunet.2014.09.003. https://arxiv.org/abs/1404.7828.

[71] Schuster, M. and Nakajima, K. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. **2012**. doi:10.1109/ICASSP.2012.6289079.

[72] Schweter, S. Berturk - bert models for turkish, **2020**. Accessed April 28, 2022. https://github.com/stefan-it/turkish-bert.

[73] Sergey Parakhin, E. S., Oleg Smirnov. How to build question answering system for online store with bert. **2020**. Accessed April 29, 2022. https://blog.griddynamics.com/question-answering-system-using-bert/.

[74] Shrestha, L. and McKeown, K. Detection of question-answer pairs in email conversations. COLING '04, page 889–es. Association for Computational Linguistics, USA, **2004**. doi:10.3115/1220355.1220483. https://doi.org/10.3115/1220355.1220483.

[75] Silge, J. Text mining of stack overflow questions, **2017**. Accessed April 30, 2022. https://stackoverflow.blog/2017/07/06/text-mining-stack-overflow-questions/.

[76] Slack. Enabling interactions with bots. Slack.com. **2021**. Accessed January 29, 2022. https://api.slack.com/bot-users#bots-overview.

[77] Sun, C., Qiu, X., Xu, Y., and Huang, X. How to fine-tune bert for text classification? *arXiv.org*, **2019**. https://arxiv.org/abs/1905.05583.

[78] Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. **2014**.

[79] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv.org*, **2017**. https://arxiv.org/abs/1706.03762.

[80] Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., and Tang, X. Residual attention network for image classification. *arXiv.org*, **2017**. https://arxiv.org/abs/1704.06904.

[81] Wang, H. and Raj, B. On the origin of deep learning. *arXiv.org*, **2017**. https://arxiv.org/abs/1702.07800.

[82] Wikipedia. Dil ailesi. Wikipedia.com. **2021**. Accessed January 29, 2022. https://tr.wikipedia.org/w/index.php?title=Dil_ailesi&stable=1.

[83] Wikipedia. Natural language processing. **2022**. Accessed January 29, 2022. https://en.wikipedia.org/wiki/Natural_language_processing.

[84] Winastwan, R. Text classification with bert in pytorch - towards data science. Towardsdatascience.com. **2021**. Accessed January 29, 2022. https://towardsdatascience.com/text-classification-with-bert-in-pytorch-887965e5820f.

[85] Young, T., Hazarika, D., Poria, S., and Cambria, E. Recent trends in deep learning based natural language processing. *arXiv.org*, **2017**. https://arxiv.org/abs/1708.02709.

[86] Zhang, X., Yang, A., Li, S., and Wang, Y. Machine reading comprehension: a literature review. *arXiv.org*, **2019**. https://arxiv.org/abs/1907.01686.