G. ALTINAY

ATILIM UNIVERSITY 2023

EMOTION ANALYSIS ON TURKISH TEXTS

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

GİZEM ALTINAY

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF INFORMATION SYSTEMS ENGINEERING

JULY 2023

EMOTION ANALYSIS ON TURKISH TEXTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

GİZEM ALTINAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS ENGINEERING

JULY 2023

Approval of the Graduate School of Natural and Applied Sciences, Atilim University.

_____

Prof. Dr. Ender KESKİNKILIÇ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Information Technologies, Atılım University**.

_____

Prof. Dr. Murat KOYUNCU
Head of Department

This is to certify that we have read the thesis EMOTION ANALYSIS ON TURKISH TEXTS submitted by GİZEM ALTINAY and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Çiğdem TURHAN
Supervisor

Prof. Dr. Murat KOYUNCU
Information Systems Engineering Department,
Atılım University _____

Assoc. Prof. Dr. Çiğdem Turhan
Software Engineering Department, Atılım University _____

Assoc. Prof. Dr. Gül Tokdemir
Computer Engineering Department, Çankaya University _____

Date: *06.07.2023*

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Gizem ALTINAY

Signature :

ABSTRACT

EMOTION ANALYSIS ON TURKISH TEXTS

Altınay, Gizem

M.Sc., Department of Information Systems Engineering

Supervisor : Assoc. Prof. Dr. Çiğdem Turhan

July 2023, 52 pages

In today's world, the amount and variety of data is increasing rapidly, and this has made data classification even more important. However, studies with Turkish texts are very limited. The purpose of this study is to classify emotions from Turkish texts and analyze the earlier research in this field. The following research questions are aimed to be answered: How is emotion analysis performed on texts? Which emotion categories can be detected? How does the BERT model perform on Turkish? This study aims to perform emotion analysis in Turkish texts with the program developed using Python language and BERT model on the Google Colab platform. The emotions classified in the study are anger, surprise, fear, happiness, love, and sadness.

Keywords: emotion analysis, Turkish language, emotion detection, text classification.

ÖZ

TÜRKÇE METİNLER ÜZERİNDE DUYGU ANALİZİ

Altınay, Gizem
Yüksek Lisans, Bilişim Sistemleri Mühendisliği
Tez Yöneticisi : Doç. Dr. Çiğdem Turhan

Temmuz 2023, 52 sayfa

Günümüz dünyasında veri miktarının ve çeşitliliğinin hızla artması veri sınıflandırmasını daha da önemli hale getirmiştir. Ancak Türkçe metinlerle yapılan çalışmalar oldukça sınırlıdır. Bu çalışmanın amacı, Türkçe metinlerden duyguları tasnif etmek ve bu alanda daha önce yapılmış araştırmaları incelemektir. Aşağıdaki araştırma sorularına cevap aranmaktadır: Metinler üzerinde duygu analizi nasıl yapılır? Hangi duygu kategorileri tespit edilebilir? BERT modeli Türkçe'de nasıl performans gösteriyor? Bu çalışma, Google Colab platformunda Python dili ve BERT modeli kullanılarak geliştirilen program ile Türkçe metinlerde duygu analizi yapmayı amaçlamaktadır. Çalışmada sınıflandırılan duygular öfke, şaşkınlık, korku, mutluluk, sevgi ve üzüntüdür.

Anahtar Kelimeler: duygu analizi, Türk dili, duygu algılama, metin sınıflandırması.

To my grandmother, Dr.Nezihe Demirsoy

ACKNOWLEDGMENTS

I would like to express my gratitude to my dear supervisor Assoc. Prof. Dr. Çiğdem Turhan, for her invaluable support throughout this thesis study.

I shall also thank to jury members Prof. Dr. Murat Koyuncu and Assoc. Prof. Dr. Gül Tokdemir for their contributions and valuable insights.

Finally, I would like to extend my heartfelt thanks to my beloved family. I am also deeply grateful to my friends who have never withheld their support from me.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

BERT	Bidirectional Encoder Representations from Transformers
NLP	Natural Language Processing

CHAPTER 1

INTRODUCTION

Emotion analysis is an analytical method used to understand and evaluate the emotion content of information sources such as texts, images, or other media sources. Emotion analysis uses an algorithm to detect emotion expressions and classify them into various emotion categories. This analysis adopts a structured approach at the word level or sentence level to detect emotional states such as anger, surprise, fear, happiness, love and sadness.

In today's world, the amount and variety of data is increasing rapidly, and this has made data classification even more important. Emotion plays a vital role in human communication, and humans have inherited the ability to feel emotions. Emotion expressions deepen the meaning of communication, strengthen emotional connections and provide interaction between individuals. However, understanding emotion from text is a difficult and complex process [1].

The emotion meaning of a word or sentence may change depending on its place in the text, its context, tone, and other linguistic features. Emotion analysis can help identify people's emotion states and attitudes, providing a valuable tool for understanding and evaluating emotion responses from texts. The basic emotion models presented by Ekman and Plutchnik [2], [3] are generally used for emotion classification.

In Emotion Analysis studies, the model is usually trained by working with large data sets, and the necessary values are reached. Since Turkish is structurally different and complex from other languages, obtaining a Turkish data set and working in this area is somewhat limited and few. As a result, since there are few emotion analysis studies in Turkish literature, emotion analysis study on Turkish texts was preferred in this study.

In this study, a structure that deals with emotion analysis on Turkish texts is emphasized and six emotion categories anger, surprise, fear, happiness, love, and sadness are examined.

The following research questions are aimed to be answered: How is emotion analysis performed on texts? Which emotion categories can be detected? How does the BERT model perform on Turkish?

This study will focus on the basic concepts of emotion analysis, methods, available techniques, and application areas. The challenges facing emotion analysis and future research directions will also be addressed.

In this study, firstly, the definition of emotion analysis is given in the introduction part. Emphasizing on emotion analysis and its importance, the missing sections in this field are also mentioned. The focus of this study is to classify texts by inferring emotions in Turkish texts. Then, in the second part, the studies done in this field are mentioned. At the same time, Turkish language structure and differences were also examined. In addition to these, the BERT model we use and its features are explained. In the third chapter, it is mentioned what kind of technique was developed to do emotion analysis on Turkish texts, the programming language used, the application developed and the environment in which it was developed. In the fourth chapter, the data obtained from the system are explained. Finally, the results of the study were explained in the conclusion section, research questions were answered and evaluations about future work were made.

CHAPTER 2

BACKGROUND INFORMATION

Emotion analysis is used to classify texts according to feeling categories. People feel emotions while communicating, or reading texts. For this reason, emotion analysis has a very important place in human life. While people can easily understand each other's feelings by their behavior, the same situation is quite complicated for texts. Through emotion analysis, it is possible to classify and extract emotions from texts according to feelings [1]. Emotion analysis and sentiment analysis are research areas that aim to automatically classify text processing and analysis [4]. In comparison with emotion analysis and sentiment analysis, sentiment analysis can only be classified in terms of positive and negative feelings, while detailed classification can be made according to feeling categories in emotion analysis. In emotion analysis, there are three main approaches: machine learning, deep learning, and lexicon-based methods to detect the emotions in order to understand human-machine interactions [5].

2.1 NLP

Natural Language Processing (NLP) is a generative mechanism used by computers and artificial intelligence to comprehend human language. Data analysis is becoming more important as a result of the growing amount of data every day. NLP has received a lot of attention in recent years for analyzing human language computationally [6]. NLP is a helpful field for researchers to analyze texts and develop an analysis path to use the data [7]. The way people understand and use language is the most important part of gathering information for researchers using NLP [8]. The goal of NLP is to make text studies as effective and accurate as a human beings [9]. Recent breakthroughs in NLP in particular have made researchers eager to develop applications for text classification operations [10].

Using NLP, machine translation, information extraction, text and speech processing, content categorization, topic discovery and modeling, contextual extraction, speech-to-text transformation, automatic summarization, and emotion analysis are possible.

In this study, BERT, Bidirectional Encoder Representation from Transformers, which is a language model developed by Google, was used for emotion analysis in a Turkish text. According to the bidirectional feature of the BERT method, it becomes easier to extract the meanings of the texts and to classify them according to words. [11].

2.2 BERT model

BERT, which stands for Bidirectional Encoder Representations of Transformers is described as a natural language processing technique that uses both artificial intelligence technologies and machine learning [12].
BERT, as a feature, evaluates the sentence by reading both from left to right and right to left, which is different from other models. It has the General Language Model structure, which is designed to be used in multiple problems such as text classification, chatbot etc. . In their study, Vaswani et al. [13] contributed to their study by publishing the tensor2tensor library for the bidirectional transformer encoder suitable for the BERT model architecture. In addition, Özçift et al. stated that by the bidirectional structure of the BERT model, fine-tuning can be done more efficiently in NLP tasks. This study published the first application in Turkish study the BERT model [14].

The BERT model includes two more techniques, Masked Language Modeling(MLM) and Next Sentence Prediction(NSP) besides being bidirectional. In MLM, the goal is to make more reliable and meaningful classifications that result in better performance using the mask structure [15]. NSP is an important part of the BERT model that allows the next sentence to be known to develop a model that can understand the relationships between sentences [16]. Devlin et al. [17] used MLM and NSP instead of using traditional BERT usage from left-to-right and right-to-left.

MLM technique is used in 15% of the words belonging to the sentence sent to the model. 80% of these words selected to use the MLM technique are exchanged with the

[MASK] token, and 10% with a random word selected. The remaining 10% is left unchanged.

In the study Arora et al. [18] conducted an experimental procedure, model evaluation explanations are divided into two parts: a traning part and and a test part in the BERT model. In these two main sections, which are similar to each other, the data is separated according to certain percentages and the BERT model working structure is executed according to the output.

In the training part, after the first sentence, a prediction is made about which sentence will follow and the system continues to process sentences in this way. Before this technique 50% of the second sentence is changed randomly then 50% is left the same.

Fine-tuning used for text classification and categorization is continued with the trained language models. Lee and Hsiang [19] performed a classification study for the patent using the pre-trained BERT model.

BERT uses the transformer structure to process the text. Figure 2.1 shows the Transformer structure.

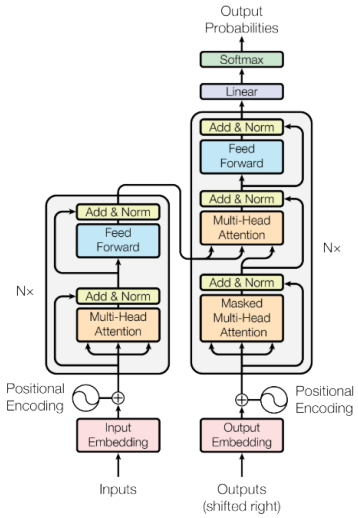

Figure 2.1 The Transformer Model Architecture [13]

As seen in Figure 2.1 above, the Transformer utilizes a similar structure in which the encoder and decoder consist of multiple layers of self-attention and fully connected layers, as depicted on the left and right sides [13].

In addition to Transformers, there are special tokens that the BERT model uses and customizes the classification process. These special tokens are described below:

1. [CLS]: CLS is a Classification Token that represents the result vector of the text to be used for tasks such as classification or labeling.

2. [SEP]: SEP stands for Seperation Token that distinguish two different sentences or text to seperate the parts.

3. [PAD]: Padding Token, used for getting equal sized input strings. PAD is especially important for multiple texts classifications.

4. [UNK]: Unknown Token is the special assigned token that if the model does not know the word.

In Figure 2.2, an example of a sentence is shown in Colab platform.

```
#text sample
sample = 'When was you play basketball? You are at home all the time!'
```

Figure 2.2 Sample Text

Then we perform the process of separating this sentence into its tokens as seen in Figure 2.3.

```
#Some basic operations can convert the text to tokens and tokens to unique integers (ids):
tokens = tokenizer.tokenize(sample)
token_ids = tokenizer.convert_tokens_to_ids(tokens)

print(f' Sentence: {sample}')
print(f'   Tokens: {tokens}')
print(f'Token IDs: {token_ids}')
```

Figure 2.3 Convert to Token

The output we get after running the code is seen in Figure 2.4.

```
 Sentence: When was you play basketball? You are at home all the time!
   Tokens: ['When', 'was', 'you', 'play', 'basketball', '?', 'You', 'are', 'at', 'home', 'all', 'the', 'time', '!']
Token IDs: [1332, 1108, 1128, 1505, 3163, 136, 1192, 1132, 1120, 1313, 1155, 1103, 1159, 106]
```

Figure 2.4 From Sentence to Token

After that, let's examine how to use private tokens.

```
#special token
#[SEP] marker for ending of a sentence

tokenizer.sep_token, tokenizer.sep_token_id
```

```
('[SEP]', 102)
```

Figure 2.5 [SEP] token and output

```
#special token
#[CLS] we have to add this token to the start of each sentence, BERT knows that we are doing classification

tokenizer.cls_token, tokenizer.cls_token_id
```

```
('[CLS]', 101)
```

Figure 2.6 [CLS] token and output

```
#special token
#[PAD] for padding

tokenizer.pad_token, tokenizer.pad_token_id
```

```
('[PAD]', 0)
```

Figure 2.7 [PAD] token and output

```
#BERT understands tokens that were in the training set
#Everything else can be encoded using the [UNK] (unknown) token:

tokenizer.unk_token, tokenizer.unk_token_id
```

```
('[UNK]', 100)
```

Figure 2.8 [UNK] token and output

In Figure 2.5, Figure 2.6, Figure 2.7, and Figure 2.8 it is seen that the code is written and output to identify special tokens.

Then, by using encode_plus() method, firstly [CLS] and [SEP] special tokens are added and then tensor is returned. In Figure 2.9, it is shown.

```
encoding = tokenizer.encode_plus(
  sample,
  max_length=32,
  add_special_tokens=True, # Add '[CLS]' and '[SEP]'
  return_token_type_ids=False,
  pad_to_max_length=True,
  return_attention_mask=True,
  return_tensors='pt',  # Return PyTorch tensors
)

encoding.keys()
```

Figure 2.9 Encoding the Tokens

After running the program, we get the output as seen in Figure 2.10 below.

```
dict_keys(['input_ids', 'attention_mask'])
```

Figure 2.10 Input Id and Attention Mask

The token ids are stored in a tensor is shown in Figure 2.11 and Figuıre 2.12.

```
print(len(encoding['input_ids'][0]))
encoding['input_ids'][0]
```

Figure 2.11 Encoding Input Id

```
tensor([ 101, 1332, 1108, 1128, 1505, 3163,  136, 1192, 1132, 1120, 1313, 1155,
        1103, 1159,  106,  102,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0])
```

Figure 2.12 Tensor for Input Id

In Figure 2.13 and 2.14, attention mask is stored.

```
print(len(encoding['attention_mask'][0]))
encoding['attention_mask']
```

Figure 2.13 Encoding Attention Mask

```
tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0]])
```

Figure 2.14 Tensor for Attention Mask

Then, by reversing the token we can look at the special tokens and their placement. In Figure 2.15 and Figure 2.16, this transformation is shown.

```
tokenizer.convert_ids_to_tokens(encoding['input_ids'][0])
```

Figure 2.15 Tokenize Convert

Figure 2.16 Special Tokens

2.3 Turkish Language

The most important factor that provides communication between people is language. Language rules vary between languages, and Turkish is a difficult language with its characteristics. Due to its agglutinative structure, Turkish has problems in natural language processing. Many different meanings of a word can be obtained by adding suffixes to words in Turkish, and this increases the feature of Turkish having a complex language structure [20]. In contrast to languages with an analytical language structure such as English, Turkish has a difficult language structure because it is a unifying language and has a complex morphology as well as a rich structure [12][14]. Syntactic and semantic exploitation of word structure are important when using applications in morphologically rich languages [21].

2.4 Related Work

As a research methodology, systematic mapping was applied and it was aimed to present an emotion analysis study on texts. A structure was created by following the steps suggested below in the systematic mapping study process [22].

- Definition of the research questions,
- Search strategy,
- Conducting a search,
- Classifying the studies,
- Data extraction

In Figure 2.2 below, the process and its steps are seen.

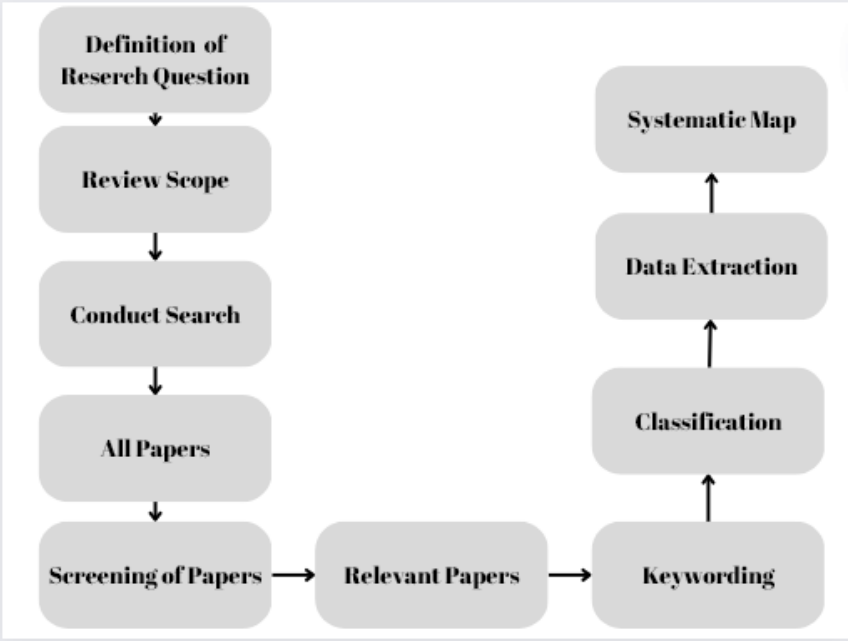

Figure 2.17 The Systematic Mapping Process

The following research questions will be answered in this study:

RQ1. How is emotion analysis performed on texts?

RQ2. Which emotion categories can be detected?

RQ3. How does the BERT model perform on Turkish?

Respective goals of research questions:

RQ1: Find the best method

RQ2: Identify the emotion from the text

RQ3: Understand the BERT model and its features

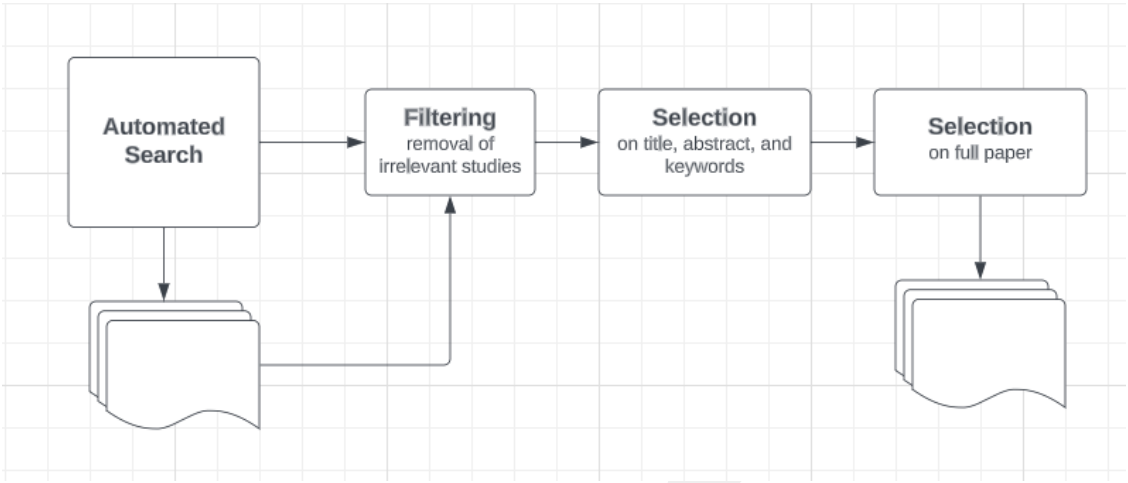

Figure 2.18 Screening for relevant papers

As seen in Figure 2.18 above, an automated search is conducted to ensure the relevant papers by using the filtering and selection processes.

For data abstraction, articles related to the subject are selected by looking at the title, keyword and abstract sections.

In Figure 2.19 and 2.20 below, according to the systematic mapping process, distribution of the primary studies and the selection process is shown.

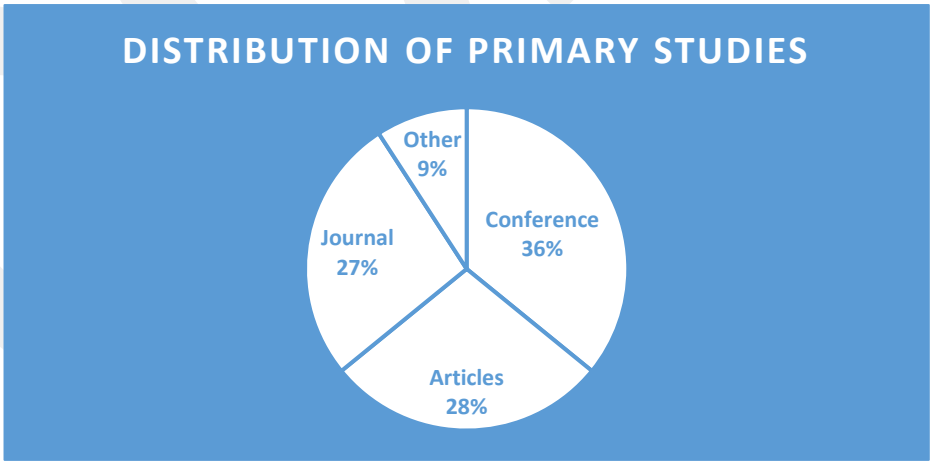

Figure 2.19 Related Studies Distribution

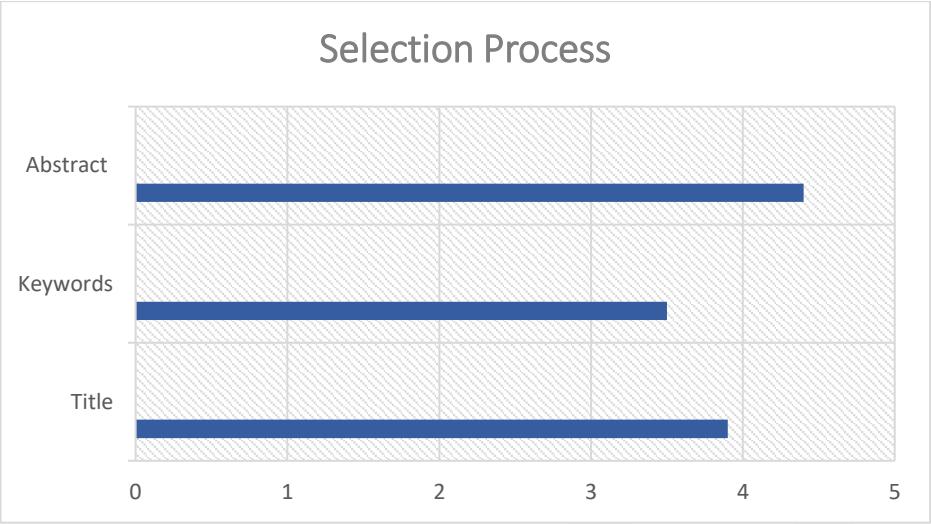

Figure 2.20 Selection Process

As a result of the systematic mapping study, it was seen that the suggested approaches to emotion analysis are on the processing of English datasets. WEKA and TREMO were used as tools in these studies. The authors conducted studies on the basic emotions of happiness, sadness, anger, fear, surprise, and disgust. For the models developed to determine these emotions, tweet sentences, social media, and e-commerce data were used as datasets. These data are processed with machine learning, deep learning, and natural language processing algorithms. Studies on the subject in the literature are presented in this section.

With the language models created for computerized linguistics, determining sentence boundaries, separating expressions, finding the roots and bodies of words, morphological analysis, semantic analysis, and summarization are possible [23].

In previous studies, there are generally more research on English as a language, and also emotion classification datasets are limited in Turkish. Scientists who want to solve the problem of subjective content changing according to context and the author's feelings, aim to identify the need to understand the feelings that expressions contain beyond their meaning . In the first studies on sentiment analysis, the aim was to examine whether the text was positive or negative [24].

As an example of studies conducted in English, Kaur and Saini [25] tried to detect emotions from English texts by using informal and formal writing styles. In this study, vector methods used as an extraction method of emotion analysis. Shrivastava et al. [26] conducted a research for detecting emotions from TV shows' transcript using CNN to focus on words to make the classification.

For Turkish texts, the first emotion analysis study was conducted by Boynukalın and Karagöz [27] on comparing different emotion categories. In their study, 80% success rate was achieved on a data set of 4000 in the categories of joy, sadness, anger, and fear.

In the study conducted by Demirci [24] for Turkish, analysis was made on tweets using the Ekman model. According to the Ekman model, there are six categories of emotions: sadness, joy, fear, anger, disgust, and surprise [28]. In addition, the Plutchik model consists of eight categories: anger, sadness, joy, disgust, fear, surprise, trust, and anticipation [2]. In their study, Toçoğlu and Alpkoçak [29] again achieved success in emotion analysis with the dictionary method over keywords and tweets using the Ekman model. In addition, Toçoglu and Alpkoçak [30] have developed a new data set called TREMO to analyze emotions from Turkish texts. The data set was created by surveying 4709 people for the six emotion categories: happiness, fear, anger, sadness, disgust, and surprise. Doğan and Kaya [31] tried to use Zemberek-NLP as an NLP library for the Turkish language to develop a root-finding process that uses stop words to determine whether the text is positive or negative.

Moreover, Yar et al. [32] explained the problem of emotion classification in texts in languages with morphologically additive language structures, such as Turkish, and how it can be solved. Moreover, by using seven emotion categories, happy, shame, guiltiness, disgust, sadness, angry, and fear, Toçoglu and Alpkoçak [33] implemented emotion classification. Also, the WEKA tool is used for validating the evaluations.

As can be seen from the studies described so far; in Turkish studies, instead of using pre-trained models, predominantly models with dictionaries are used.

Distinctly, in English, even if a suffix is added to words, the semantic structure remains almost the same, so it is easier to apply the pre-trained models, and more accurate results are achieved.

In their study, Devlin et al. [17] investigated the BERT model. According to this study, the pre-trained language models, BERT, has two sides: feature-based and fine-tuning.

In another study, Celikten, and Bulut conducted medical text classification using BERT and English data sets. The reasons for doing their work with the English dataset were that Turkish has a difficult morphological structure for natural language processing, and they stated that the research in Turkish was limited especially in the medical field [34]. Kannan and Kothamasu [35] conducted a study using BERT to categorize tweets, also they used related libraries and transformers that ease the classification of the texts.

Abas et al. [36] conducted a study about BERT-CNN that includes three parts: pre-processing the data, BERT-based model, and CNN which is used as a classifier.

Ein-Dor et al. [37] worked with BERT to bring different structures to classical and limited studies in the field of Active Learning (AL). Furthermore, it is aimed to perform an analysis for comparing the AL strategies to ensure improved future work.

In addition to these studies, although in small quantities, there are some studies that exist for emotion analysis in Turkish texts with pre-trained models such as BERT.

Özdil et al. [38] conducted a study for classification of online marketing platforms using the BERT model for Turkish. In the study based on online advertising platforms, the texts were automatically classified using the NLP-based method. Furthermore, Aygün et al. [39] used BERT model for classification of the tweets written in the period of pandemic. Çatal and Nangir [40] have done a classification study in Turkish by proposing a new method with multiple classification systems for Turkish. Aytan and Sakar [41] used Turkish emotion analysis for text classification using the RoBERTa model which is a variation of the BERT model.

Othan et al. [42] conducted a study for Turkish using BERT to ensure valuable contribution to classification in the texts. In this study, classification and analysis of financial texts were made.

Köksal et al. [9] have done category classification for e-commerce systems. In their studies, text classification and user intent analysis models were developed on the basis of e-commerce product category using pre-trained libraries in Turkish with BERT, ELECTRA and RoBERTa. Okur and Sertbaş [43] used the BERT model to classify Turkish texts. They focused on machine learning and machine translation using ready-made libraries and natural language processing structure. In addition, Karayiğit et al. [44] conducted a study to analyze Turkish Instagram Comments using the BERT model.

CHAPTER 3

DESIGN AND IMPLEMENTATION

This study aims to perform emotion analysis in Turkish texts with the program developed using Python language and BERT model on Google Colab platform. The emotions classified in the study are anger, surprise, fear, happiness, love and sadness.

3.1. Dataset

First of all, Turkish datasets that can be used in this field were investigated since the purpose of this study is to classify Turkish texts in terms of emotion. As a result of this investigation, it was seen that Turkish studies in this field are limited and there are very few Turkish data sets. First of all, this study was conducted with a limited number of data sets prepared. Considering the contribution of working with more data to the results, it was decided that using the Turkish version of the dataset prepared in English would give more efficient and accurate results. Due to these limitations, the dataset, which was prepared in English for emotion analysis on the Twitter platform, was first translated into Turkish. Then, it was brought to the required format to be used in the system. First, some data were adapted to the format, and scores and values were calculated. Then, by increasing the data, the score and the required values were provided to give more successful results. The data set created for the study includes emotions and sentences consisting of categories and text parts. First of all, the category section was determined to process according to the emotion categories, then the Turkish texts and the emotion contained in these texts were determined.

In this study, fine-tuning technique was used to solve the text classification problem and dbmdz/bert-base-turkish-128k-uncased ready-made model from Hugging Face library was used to make emotion analysis for Turkish texts. If the steps of the fine-tuning technique are mentioned:

1. Selection of the Pre-Trained Model: A suitable pre-trained model should be selected for the task of interest. This model can be trained to learn general language meanings and structures, often on a large data set. For example, models such as BERT, or ROBERTa can be used. This study was conducted with BERT.
2. Dataset Preparation
3. Model Arrangements: Modifying the pre-trained model according to the task to be performed or adding some layers.
4. Fine-tuning Process

3.2. Program Development Phase

In this section, the program and its development stages will be explained step by step.

First, we create our .ipynb workspace, which we will write with Python on Google Colab. Right after, we connect our Google Drive account to our work so that we can access the necessary files while writing our program. After getting our workspace ready, we change the runtime to GPU to get better performance as accelerator. (Colab -> Runtime -> Change Runtime Type -> GPU)

The reason for using GPU is to write and compile our codes in a faster and more active working environment.

After these installations and settings, now, the working environment is ready and the programming steps can be explained.

First, the necessary modules are imported into the project as seen in the figure below.

```
import pandas as pd
import tensorflow as tf
import torch
import numpy as np
import time
import datetime
import random
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from google.colab import drive

!pip install transformers
import transformers
from transformers import BertTokenizer
from torch.utils.data import TensorDataset, random_split
from torch.utils.data import DataLoader, RandomSampler, SequentialSampler
from transformers import BertForSequenceClassification, AdamW, BertConfig
from transformers import get_linear_schedule_with_warmup

from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
```

Figure 3.1 Modules

If we explain the important modules:

- tensorflow: can be expressed as deep learning library that contains open source codes.
- transformers: it is a structure that allows to create a bond by examining the meanings between words.

After this process, transformers required for BERT and libraries required for Python are made available in the system.

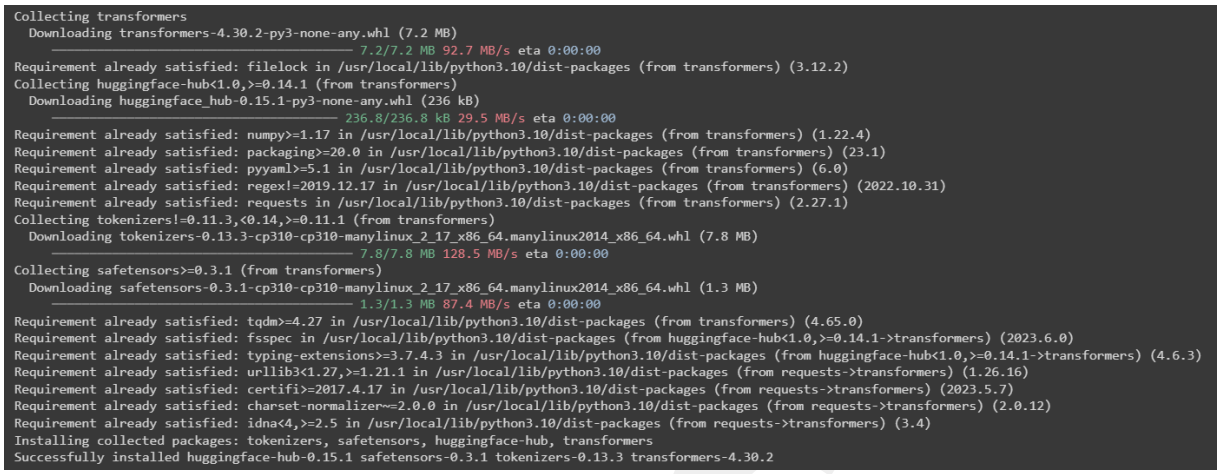

Figure 3.2 Installation Output

In Figure 3.2, it is clearly seen that transformers, huggingface library, tokenizers, tensors are successfully uploaded. After that, we have to check the GPU as seen in Figure 3.3.

```
# check GPU
device_name = tf.test.gpu_device_name()
if device_name == '/device:GPU:0':
    device = torch.device("cuda")
    print('GPU:', torch.cuda.get_device_name(0))
else:
    raise SystemError('GPU device not found')
```

```
GPU: Tesla V100-SXM2-16GB
```

Figure 3.3 GPU Checking

Then, the environmet is set as Google Drive to the folder that will be used as a data path.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; t
```

Figure 3.4 Google Drive Connection

```
data = pd.read_csv(data_path + 'GABertData.csv')
data['encoded_categories'] = LabelEncoder().fit_transform(data['category'])

print(data.sample(15))
print(data.groupby('category').size())
```

Figure 3.5 Encoded Categories

Using read_csv() function the dataset is ensured and the system can read the data from this path. As seen in Figure 3.5, the category column needs to be encoded so that labels can be used categorically in the model. After this process, as can be seen in the figures below, the output in which the emotions are categorized is obtained from our data set.

```
        category                                               text  \
37         anger    Yanında argo bir dille konuştuğu için ona bağ...
93     happiness    yazları Mersin'de esans kıvamında kokan porta...
158      sadness    yaşlı bir çiftin el ele yürürken aniden gözle...
150      sadness    Kaybettiğimiz sevdiklerimizi anımsadığımda ka...
107    happiness      Dünyayı daha güzel bir yer haline getirebilmek
129     surprise                          Yeni bir yemek deneyimi
202         love     aile gibi arkadaşlara sahip olmanın verdiği his
111    happiness    Sevdiklerimle istediğim zaman istediğim yere ...
62          fear           Sevdiklerinin yanında olamamaktan gerilme
2          anger    Sebebini sorarken yüksek sesle ve sinirli bir...
190         love    Seninle birlikteyken içimde bir kıpırtı ve he...
106    happiness                    İhtiyacı olanlara yardım etmek
72          fear          doğal afetlere karşı savunmasız hissetmek
182         love    Seni sevmek benim için en büyük mutluluk kaynağı
4          anger    Konuşurken yüksek sesle ve gergin bir şekilde...
```

Figure 3.6 Processing Emotion Categories into Sentences

```
     encoded_categories
37                    0
93                    2
158                   4
150                   4
107                   2
129                   5
202                   3
111                   2
62                    1
2                     0
190                   3
106                   2
72                    1
182                   3
4                     0
```

Figure 3.7 Encoded Emotion Categories

As seen in Figure 3.6 and Figure 3.7, first of all, emotion categories are assigned to 15 randomly selected sentences from the data set. Afterwards, the categories are checked and information is given about how many of each of them are present.

Then, the model and tokenizer required for BERT are downloaded, as can be seen in the figure below.

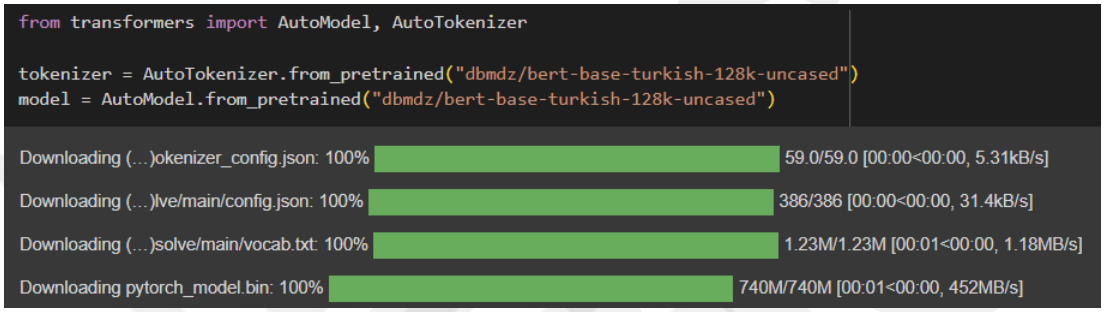

Figure 3.8 BERT Tokenizer and Model

dbmdz/bert-base-turkish-128k-uncased is a Turkish model that includes config.json, pytorch_model.bin and vocab.txt to classify the Turkish texts using BERT model. In Figure 3.8, it is loaded, with transformers BERT uncased model.

Figure 3.9 shows that training and test data are divided into two parts, 80% for training, and 20% as test data.

```
training = data.groupby('category').apply(lambda x : x.sample(frac = 0.8))
test = pd.concat([data,training]).drop_duplicates(keep=False)

print("Training: ", len(training))
print("Test: ", len(test))

training_texts = training.text.values
training_labels = training.encoded_categories.values
```

Figure 3.9 Training and Test Data

In order to use the text data in the model, we first need to process it. The sentence in our text is first separated into its tokens with the tokenizer process, and the tokens required for the classification process are added to the beginning and end of the sentence. Since the input vector is of fixed length, if the analyzed sentence is shorter than the maximum length, the remaining blanks are filled. Then the attention masks are created and the tensor object is created. In the output obtained, first the sentence is divided into tokens and then divided into index numbers. All this can be seen in the Figure 3.10 and Figure 3.11 below.

```
input_ids = []
attention_masks = []

for text in training_texts:
    encoded_dict = tokenizer.encode_plus(
                        text,
                        add_special_tokens = True,
                        max_length = max_len,
                        pad_to_max_length = True,
                        return_attention_mask = True,
                        return_tensors = 'pt',
                   )

    input_ids.append(encoded_dict['input_ids'])
    attention_masks.append(encoded_dict['attention_mask'])

input_ids = torch.cat(input_ids, dim=0)
attention_masks = torch.cat(attention_masks, dim=0)
labels = torch.tensor(training_labels)

print('Original: ', training_texts[0])
print('Token IDs:', input_ids[0])
```

Figure 3.10 Input Tokenizer

```
Original:   Yalan söylemelerine karşı kayıtsız kalamayacağını sert bir şekilde ifade etti
Token IDs: tensor([     2,   5238,  83195,   4165,  40209,  20767,  47818,   1984, 123433,
          4992,   1947,  18011,   3017,   2517,      3,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0,      0,      0,      0])
```

Figure 3.11 Token Ids of The Sentence

As can be seen, the words in the text give us an output by expressing the ids of the tokenizer.

```
train_dataset = TensorDataset(input_ids, attention_masks, labels)

batch_size = 32

train_dataloader = DataLoader(
            train_dataset,
            sampler = RandomSampler(train_dataset),
            batch_size = batch_size
        )

number_of_categories = len(data['encoded_categories'].unique())

model = BertForSequenceClassification.from_pretrained(
    "dbmdz/bert-base-turkish-128k-uncased",
    num_labels = number_of_categories,
    output_attentions = False,
    output_hidden_states = False,
)

model.cuda()
```

Figure 3.12 Data Loader

The created tensor data needs to be processed into the model and by creating a variable called dataloader, the pre-trained model is created with the fine-tuning method, as is done in the tokenizer process as seen in Figure 3.12 above.

In Figure 3.13 below, the output showing that the bert model and its requirements have been successfully installed and can be used.

```
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(128000, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=6, bias=True)
)
```

Figure 3.13 Bert Model Output

```
epochs = 4

optimizer = AdamW(model.parameters(),
                  lr = 5e-5,
                  eps = 1e-8
                )

total_steps = len(train_dataloader) * epochs
scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps = 0,
                                            num_training_steps = total_steps)
```

Figure 3.14 Optimization Phase

Before the training, the learning rate is optimized using Adam Optimizer to finally establish a more efficient structure.

Adam Optimizer, which was also used by Kohli et al. [45], was used for the optimization phase. Adam Optimizer combines the good features of algorithms and ensures the successful execution of the optimization phase even in problematic situations. The training phase is done 4 times for our program, as seen in Figure 3.14.

CHAPTER 4

EVALUATION

In this section, the evaluation of the emotion analysis is performed. The details about the aim of the study, the methods used and the dataset used were presented in the previous sections. The results of the experiments, the performance of the model and the evaluation of the results obtained will now be presented.

The epoch number represents the number of times the dataset is passed while the emotion analysis is being performed in the system. In Figure 4.1 below, the output shows how long every 15-input took, how long it took to complete a training section, and the average loss values at the end of each section can be examined.

As seen in Figure 4.1, before proceeding to the training phase, we fix the seed value so that the same results can be achieved in all trials.

```python
def format_time(elapsed):
    elapsed_rounded = int(round((elapsed)))
    return str(datetime.timedelta(seconds=elapsed_rounded))

seed_val = 3333

random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)

training_stats = []
total_t0 = time.time()

for epoch_i in range(0, epochs):
    print('======== Epoch {:} / {:} ========'.format(epoch_i + 1, epochs))
    t0 = time.time()
    total_train_loss = 0
    model.train()

    for step, batch in enumerate(train_dataloader):
        if step % 10 == 0 and not step == 0:
            elapsed = format_time(time.time() - t0)
            print('Batch {:>5,}  of  {:>5,}.    Elapsed: {:}.'.format(step, len(train_dataloader), elapsed))

        b_input_ids = batch[0].to(device)
        b_input_mask = batch[1].to(device)
        b_labels = batch[2].to(device)
```

```python
        model.zero_grad()
        output = model(b_input_ids,
                       token_type_ids=None,
                       attention_mask=b_input_mask,
                       labels=b_labels)
        loss = output['loss']
        logits = output['logits']
        total_train_loss += loss.item()
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
        scheduler.step()

    avg_train_loss = total_train_loss / len(train_dataloader)
    training_time = format_time(time.time() - t0)

    print("Average training loss: {0:.2f}".format(avg_train_loss))
    print("Training epoch took: {:}".format(training_time))

    training_stats.append(
        {
            'epoch': epoch_i + 1,
            'Training Loss': avg_train_loss,
            'Training Time': training_time,
        }
    )

print("Training completed in {:} (h:mm:ss)".format(format_time(time.time()-total_t0)))
```

Figure 4.1 Fixed Seed Value

The training phase is started by taking the inputs 32 by 32 and feeding to the model, and each segment is optimized before it starts. In the test phase, eval is used as a method and the values in the dataloader are transferred to the GPU. At the same time, the guard values are reset and the logit values are created.

```
======== Epoch 1 / 4 ========
Batch    10  of     33.    Elapsed: 0:00:04.
Batch    20  of     33.    Elapsed: 0:00:08.
Batch    30  of     33.    Elapsed: 0:00:12.
Average training loss: 1.62
Training epoch took: 0:00:13
======== Epoch 2 / 4 ========
Batch    10  of     33.    Elapsed: 0:00:04.
Batch    20  of     33.    Elapsed: 0:00:08.
Batch    30  of     33.    Elapsed: 0:00:12.
Average training loss: 1.19
Training epoch took: 0:00:13
======== Epoch 3 / 4 ========
Batch    10  of     33.    Elapsed: 0:00:04.
Batch    20  of     33.    Elapsed: 0:00:08.
Batch    30  of     33.    Elapsed: 0:00:12.
Average training loss: 0.87
Training epoch took: 0:00:13
======== Epoch 4 / 4 ========
Batch    10  of     33.    Elapsed: 0:00:04.
Batch    20  of     33.    Elapsed: 0:00:08.
Batch    30  of     33.    Elapsed: 0:00:12.
Average training loss: 0.68
Training epoch took: 0:00:13
Training completed in 0:00:53 (h:mm:ss)
```

Figure 4.2 Epoch of the System

```
data_stats = pd.DataFrame(data=training_stats)
plt.plot(data_stats['Training Loss'], label="Training")
plt.title("Training Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.xticks([1, 2, 3, 4])
plt.show()
```

Figure 4.3 Graphing the average loss

In Figure 4.4, the loss performance can be visualized to examine the model performance in training.

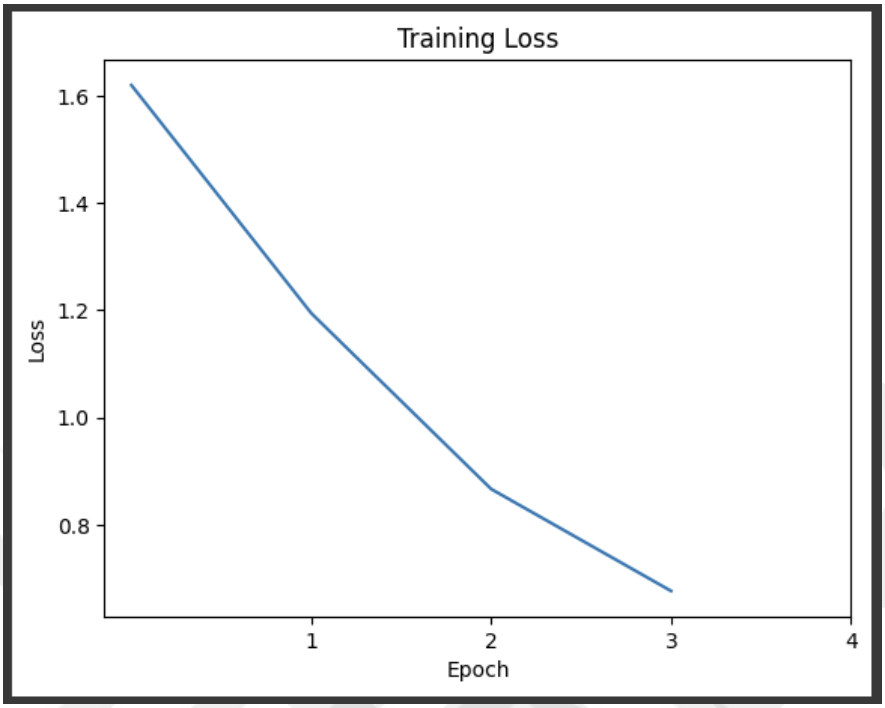

Figure 4.4 Loss Graph

```
test_texts = test.text.values
test_labels = test.encoded_categories.values

input_ids = []
attention_masks = []

for text in test_texts:
    encoded_dict = tokenizer.encode_plus(
                        text,
                        add_special_tokens = True,
                        max_length = max_len,
                        pad_to_max_length = True,
                        return_attention_mask = True,
                        return_tensors = 'pt',
                   )

    input_ids.append(encoded_dict['input_ids'])
    attention_masks.append(encoded_dict['attention_mask'])

input_ids = torch.cat(input_ids, dim=0)
attention_masks = torch.cat(attention_masks, dim=0)
labels = torch.tensor(test_labels)

batch_size = 32

prediction_data = TensorDataset(input_ids, attention_masks, labels)
prediction_sampler = SequentialSampler(prediction_data)
prediction_dataloader = DataLoader(prediction_data, sampler=prediction_sampler, batch_size=batch_size)
```

Figure 4.5 Test Phase Dataloader

A dataloader is created for the test dataset as in the training dataset, and the same operations are performed for the test phase, as seen in Figure 4.5 above.

```
print('Prediction started on test data')
model.eval()
predictions , true_labels = [], []

for batch in prediction_dataloader:
  batch = tuple(t.to(device) for t in batch)
  b_input_ids, b_input_mask, b_labels = batch

  with torch.no_grad():
      outputs = model(b_input_ids, token_type_ids=None,
                      attention_mask=b_input_mask)

  logits = outputs[0]
  logits = logits.detach().cpu().numpy()
  label_ids = b_labels.to('cpu').numpy()

  predictions.append(logits)
  true_labels.append(label_ids)

print('Prediction completed')

prediction_set = []

for i in range(len(true_labels)):
  pred_labels_i = np.argmax(predictions[i], axis=1).flatten()
  prediction_set.append(pred_labels_i)

prediction_scores = [item for sublist in prediction_set for item in sublist]
```

Figure 4.6 Prediction Phase

As seen in Figure 4.6, the test data is used to predict the results of the model. Since our batch value is 32, we process 32 inputs to the model as in the training part. The flatten function ensures that the results are collected in a single list and kept in the prediction_set variable; thus, our prediction process is realized.

Performance metrics are used to understand how successfully the model performs the emotion analysis task. Often, one metric alone does not provide an adequate assessment, so it is important to evaluate multiple metrics together. If performance metrics are mentioned, accuracy, precision, recall and f1-score metrics are used. Accuracy, expresses the ratio of correct guesses to the total number of samples.

Precision, refers to the ratio of samples predicted to be positive to samples that are actually positive. Recall, indicates how many of the truly positive samples were correctly predicted as positive. F1-score is a metric used to strike a balance between precision and sensitivity.

```
f_score = f1_score(test_labels, prediction_scores, average='macro')
precision = precision_score(test_labels, prediction_scores, average='macro')
recall = recall_score(test_labels, prediction_scores, average='macro')

print("F-Score: ", f_score)
print("Recall: ", recall)
print("Precision: ", precision)

report = pd.DataFrame(classification_report(test_labels, prediction_scores, output_dict=True))
report = report.rename(columns={'0':'anger',
                          '1':'surprise',
                          '2':'fear',
                          '3':'happiness',
                          '4':'love',
                          '5':'sadness'})

print(report)
```

Figure 4.7 Calculation of Metrics

Then, we observe the performance of the model by subtracting the precision, recall and f-score values, for the emotion categories, anger, surprise, fear, happiness, love and sadness, as seen in Figure 4.7. In the studies, it was seen that the scores obtained when using fewer datasets give better results when the dataset is increased.

While performing f1-score and precision calculations with the dataset, it has been observed that precision and f1-score values increase if the data is increased and the calculations are made again accordingly.

```
F-Score:  0.6241762568661594
Recall:  0.6127060990796457
Precision:  0.776445001839489
              anger   surprise       fear  happiness       love    sadness  \
precision  0.615385   0.764706   0.774194   0.833333   0.671053   1.000000
recall     0.615385   0.787879   0.837209   0.555556   0.796875   0.083333
f1-score   0.615385   0.776119   0.804469   0.666667   0.728571   0.153846
support   39.000000  33.000000  86.000000  27.000000  64.000000  12.000000

           accuracy   macro avg  weighted avg
precision  0.724138    0.776445      0.740472
recall     0.724138    0.612706      0.724138
f1-score   0.724138    0.624176      0.709851
support    0.724138  261.000000    261.000000
```

Figure 4.8 F-Score, Recall, Precision, Accuracy Values

Finally, as seen in Figure 4.8, the classification process according to categories can be examined. If we look at the performances, it is seen that the best performance is achieved in the fear category.

CHAPTER 5

CONCLUSION

In this article, emotion analysis in Turkish texts and its importance are examined. The importance of emotion analysis in Turkish texts emerges in many areas. In the business world, emotion analysis in texts such as customer satisfaction reviews and product reviews is an important tool for understanding brand image and product development. Emotion analysis is also used in areas such as social media analysis, public opinion surveys, and evaluation of the impact of policies.

The results of this study shows that an extensive data analysis process was carried out to answer the research questions stated at the beginning. The first research question is "How is emotion analysis performed on texts?" In the previous sections, how emotion analysis is done on texts and which techniques are used are explained in detail. The second research question is "Which emotion catgeories can be detected?" and according to the study six emotions are detected, these are, anger, surprise, fear, happiness, love and sadness in the evaluation section. The third research question is "How does the BERT model perform on Turkish?" In this study BERT model is used and also NLP techniques are discussed in detail in Chapter 2.

Performing emotion analysis in Turkish texts involves some difficulties and complexities. Factors such as the structural features of language, polysemy, irony, and the way language expresses emotional expressions can make an accurate emotion analysis difficult. Therefore, the selection of appropriate methods and models is important to obtain accurate results.

This study deals with research on the application of the BERT model to perform emotion analysis on Turkish texts. The literature survey indicates that none of the previous studies conducted emotion analysis in Turkish using the BERT model.

BERT's pre-trained language models have been trained on a large dataset to understand the general language meaning and have achieved great success in language processing.

The BERT model is fine-tuned on a dataset labeled with emotion labels. The pre-trained network of the model, which initially includes general language abilities, was customized to identify emotional expressions in Turkish texts. The fine-tuning process was performed using gradient descent and backpropagation algorithm and its performance on the validation dataset was monitored.

In conclusion, this study showed that the BERT model is an effective tool in the field of emotion analysis in Turkish texts. The high performance of the model highlights the importance of using Turkish texts in NLP for emotion analysis. Future studies could further examine the performance of the BERT model on different datasets and tasks and focus on further development of the model.

In order to advance the field of emotion analysis on Turkish texts, future research should focus on enhancing the methodologies employed. It can be successfully applied in various fields by increasing the categories of the Turkish emotion dataset and using different deep learning methods.

REFERENCES

[1] M. A. Tocoglu, O. Ozturkmenoglu, and A. Alpkocak, "Emotion Analysis from Turkish Tweets Using Deep Neural Networks," *IEEE Access*, vol. 7, pp. 183061–183069, 2019.

[2] P. Ekman, "An Argument for Basic Emotions," *Cogn. Emot.*, vol. 6, no. 3–4, pp. 169–200, May 1992.

[3] A. Surikov and E. Egorova, "Emotional Analysis of Russian Texts Using Emojis in Social Networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2021, vol. 12602 LNCS.

[4] N. Azam, B. Tahir, M. Amir, and M. Al-Khawarizmi, "Sentiment and Emotion Analysis of Text: A Survey on Approaches and Resources," *Researchgate.Net*, no. February, pp. 87–94, 2020.

[5] V. Jain and M. Parmar, "A Review on Emotion and Sentiment analysis Using Learning Techniques," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 11, 2022.

[6] M. A. A. S. Ali, "AI-Natural Language Processing (NLP)," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 9, no. VIII, 2021.

[7] R. Egger and E. Gokce, "Natural Language Processing (NLP): An Introduction," in *Applied Data Science in Tourism*, 2022, pp. 307–334.

[8] Z. A. Guven and M. O. Unalir, "Natural language based analysis of SQuAD: An analytical approach for BERT," *Expert Syst. Appl.*, vol. 195, 2022.

[9] D. Koksal, M. M. Alacan, E. Olgun, and C. O. Sakar, "Natural Language Processing-Based Product Category Classification Model for E-Commerce," in *2022 30th Signal Processing and Communications Applications Conference, SIU 2022*, 2022, pp. 1–4.

[10] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information (Switzerland)*, vol. 10, no. 4. 2019.

[11] B. Selvakumar and B. Lakshmanan, "Sentimental analysis on user's reviews using BERT," *Mater. Today Proc.*, vol. 62, 2022.

[12] M. Özkan and G. Kar, "Multiclass Classification of Scientific Texts Written in Turkish by Applying Deep Learning Technique," *Mühendislik Bilim. ve Tasarım Derg.*, vol. 10, no. 2, 2022.

[13] A. Vaswani *et al.*, "Attention Is All You Need," Jun. 2017, [Online]. Available: http://arxiv.org/abs/1706.03762

[14] A. Özçift, K. Akarsu, F. Yumuk, and C. Söylemez, "Advancing natural language processing (NLP) applications of morphologically rich languages with bidirectional encoder representations from transformers (BERT): an empirical case study for Turkish," *Automatika*, vol. 62, no. 2, pp. 226–238, 2021.

[15] P. Behnamghader, H. Zakerinia, and M. S. Baghshah, "MG-BERT: Multi-Graph Augmented BERT for Masked Language Modeling," in *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing, Mexico City, Mexico*, 2021, pp. 125–131.

[16] A. Alajrami and N. Aletras, "How does the pre-training objective affect what large language models learn about linguistic properties?," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2022, vol. 2.

[17] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of NAACL-HLT 2019*, 2019, pp. 4171–4186. [Online]. Available: https://github.com/tensorflow/tensor2tensor

[18] S. Arora, D. Pruthi, N. Sadeh, W. W. Cohen, Z. C. Lipton, and G. Neubig, "Explain, Edit, and Understand: Rethinking User Study Design for Evaluating Model Explanations," in *Proceedings of the 36th AAAI Conference on Artificial Intelligence, AAAI 2022*, 2022, vol. 36.

[19] J. S. Lee and J. Hsiang, "Patent classification by fine-tuning BERT language model," *World Pat. Inf.*, vol. 61, Jun. 2020.

[20] K. Tohma and Y. Kutlu, "Challenges Encountered in Turkish Natural Language Processing Studies," *Nat. Eng. Sci.*, vol. 5, no. 3, pp. 204–211, Nov. 2020.

[21] H. Sak, T. Güngör, and M. Saraçlar, "Resources for Turkish morphological processing," *Lang. Resour. Eval.*, vol. 45, no. 2, pp. 249–261, May 2011.

[22] E. E. Chotim and A. Pramanti, "E-Voting systems to prevent conflicts caused by false results in Elections in Indonesia," *Int. J. Innov. Creat. Chang.*, vol. 12, no. 3, pp. 508–517, 2020.

[23] A. Uçan and E. Akçapinar Sezer, "Emotion analysis in turkish computational linguistics studies," *Bellet. Yearb. Turkic Stud.*, vol. 2020, no. 70, pp. 193–210, 2020.

[24] G. M. Demirci, S. R. Keskin, and G. Dogan, "Sentiment Analysis in Turkish with Deep Learning," *Proc. - 2019 IEEE Int. Conf. Big Data, Big Data 2019*, pp. 2215–2221, 2019.

[25] J. Kaur and J. R. Saini, "Emotion Detection and Sentiment Analysis in Text Corpus: A Differential Study with Informal and Formal Writing Styles," *Int. J. Comput. Appl.*, vol. 101, no. 9, 2014.

[26] K. Shrivastava, S. Kumar, and D. K. Jain, "An effective approach for emotion detection in multimedia text data using sequence based convolutional neural network," *Multimed. Tools Appl.*, vol. 78, no. 20, 2019.

[27] Z. Boynukalin, “Emotion analysis of turkish texts by using machine learning methods,” Master Thesis, Computer Engineering Department, Middle East Technical University, Ankara, Turkey, 2012.

[28] A. Kolmogorova and A. Kalinin, “Conceptualization of time in internet-texts of different emotional tonality,” *SHS Web Conf.*, vol. 88, 2020.

[29] M. A. Tocoglu and A. Alpkocak, “Emotion extraction from turkish text,” *Proc. - 2014 Eur. Netw. Intell. Conf. ENIC 2014*, pp. 130–133, 2014.

[30] M. A. Tocoglu and A. Alpkocak, “TREMO: A dataset for emotion analysis in Turkish,” *J. Inf. Sci.*, vol. 44, no. 6, pp. 848–860, 2018.

[31] E. Dogan and B. Kaya, “Deep Learning Based Sentiment Analysis and Text Summarization in Social Networks,” *2019 Int. Conf. Artif. Intell. Data Process. Symp. IDAP 2019*, 2019.

[32] E. Yar, I. Delibalta, L. Baruh, and S. S. Kozat, “Online text classification for real life tweet analysis | Gerçek Hayat Tweet Analizi için Çevrimiçi Metin Siniflandirmasi,” *2016 24th Signal Process. Commun. Appl. Conf. SIU 2016 - Proc.*, 2016.

[33] M. A. Toçoglu and A. Alpkoçak, “Lexicon-based emotion analysis in Turkish,” *Turkish J. Electr. Eng. Comput. Sci.*, vol. 27, no. 2, pp. 1213–1227, 2019.

[34] A. Celikten and H. Bulut, “Turkish Medical Text Classification Using BERT,” in *Turkish medical text classification using BERT,” Jul. 2021*, Jun. 2021, pp. 1–4.

[35] E. Kannan and L. A. Kothamasu, “Fine-Tuning BERT Based Approach for Multi-Class Sentiment Analysis on Twitter Emotion Data,” *Ing. des Syst. d’Information*, vol. 27, no. 1, pp. 93–100, Feb. 2022.

[36] A. R. Abas, I. Elhenawy, M. Zidan, and M. Othman, “Bert-cnn: A deep learning model for detecting emotions from text,” *Comput. Mater. Contin.*, vol. 71, no. 2, pp. 2943–2961, 2022.

[37] L. Ein-Dor *et al.*, “Active Learning for BERT: An Empirical Study,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020, pp. 7949–7962.

[38] U. Ozdil, B. Arslan, D. E. Tasar, G. Polat, and S. Ozan, “Ad Text Classification with Bidirectional Encoder Representations,” in *6th International Conference on Computer Science and Engineering (UBMK)*, Sep. 2021, pp. 169–173.

[39] I. Aygun, B. Kaya, and M. Kaya, “Aspect Based Twitter Sentiment Analysis on Vaccination and Vaccine Types in COVID-19 Pandemic With Deep Learning,” *IEEE J. Biomed. Heal. Informatics*, vol. 26, no. 5, pp. 2360–2369, May 2022.

[40] C. Catal and M. Nangir, “A sentiment classification model based on multiple classifiers,” *Appl. Soft Comput. J.*, vol. 50, pp. 135–141, Jan. 2017.

[41] B. Aytan and C. O. Sakar, "Comparison of Transformer-Based Models Trained in Turkish and Different Languages on Turkish Natural Language Processing Problems," *2022 30th Signal Process. Commun. Appl. Conf.*, pp. 1–4, May 2022.

[42] Z. H. K. and M. U. Derya Othan, "Financial Sentiment Analysis for Predicting Direction of Stocks using Bidirectional Encoder Representations from Transformers (BERT) and Deep Learning Models," *17th ISTANBUL-TURKEY Int. Conf. "Innovative Intell. Technol. Istanbul*, Dec. 2019, pp. 1–5.

[43] H. I. Okur and A. Sertbas, "Pretrained Neural Models for Turkish Text Classification," in *2021 6th International Conference on Computer Science and Engineering (UBMK)*, Sep. 2021, pp. 174–179.

[44] H. Karayiğit, A. Akdagli, and Ç. İ. Acı, "BERT-based Transfer Learning Model for COVID-19 Sentiment Analysis on Turkish Instagram Comments," *Inf. Technol. Control*, vol. 51, no. 3, 2022.

[45] H. Kohli, J. Agarwal, and M. Kumar, "An improved method for text detection using Adam optimization algorithm," *Glob. Transitions Proc.*, vol. 3, no. 1, 2022.