

**MEASURING THE SENTIMENT EFFECTS USING EMOTICON FEATURES
FOR A GENERAL TURKISH CORPUS**
(EMOJİ İKONLARININ ÖZELLİKLERİ KULLANILARAK GENEL TÜRKÇE DERLEM
ÜZERİNDE DUYGU ANALİZİNİN ÖLÇÜLMESİ)

by

Çağatay Ünal YURTÖZ, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

October 2019

This is to certify that the thesis entitled

**MEASURING THE SENTIMENT EFFECTS USING EMOTICON FEATURES
FOR A GENERAL TURKISH CORPUS**

prepared by **Çağatay Ünal YURTÖZ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering** at the **Galatasaray University** is approved by the

Examining Committee:

Assist. Prof. İsmail Burak PARLAK (Supervisor)
Department of Computer Engineering
Galatasaray University -----

Assist. Prof. Murat AKIN
Department of Computer Engineering
Galatasaray University -----

Assist. Prof. Cemal Okan ŞAKAR
Department of Computer Engineering
Bahçeşehir University -----

Date: -----

ACKNOWLEDGEMENTS

I want to express my gratitude to my supervisor, Assoc. Prof. İ. Burak Parlak, for his support and guidance.

I am very thankful to my family for their encouragement and trust in me.

I would like to thank all my friends for their opinions, supports and valuable ideas.

October 2019

Çağatay Ünal YURTÖZ

TABLE OF CONTENTS

| | |
|---|-------------|
| TABLE OF CONTENTS | iv |
| LIST OF SYMBOLS | vi |
| LIST OF FIGURES | vii |
| LIST OF TABLES | viii |
| ABSTRACT | x |
| RÉSUMÉ | xi |
| ÖZET | xii |
| 1. INTRODUCTION | 1 |
| 1.1 Aim and Guidance | 3 |
| 2. LITERATURE REVIEW | 5 |
| 2.1 Sentiment Analysis Studies on Turkish Texts | 5 |
| 2.2 Emoji Based Studies | 8 |
| 3. METHODOLOGY | 10 |
| 3.1. The Corpus | 10 |
| 3.1.2 Definition of the Emoji Set..... | 11 |
| 3.1.3 Gathering the Data..... | 14 |
| 3.1.4 Review of Dataset..... | 15 |
| 3.1.4.1 Removing Incoherent Data | 15 |
| 3.1.5 Manually Labeled Data | 16 |
| 3.1.6 Final Dataset | 17 |
| 3.1.6.1 Sample Tweets | 17 |
| 3.2 Preparation | 18 |
| 3.2.1 Cleaning and Lowercasing | 19 |
| 3.2.2 Word Correction | 19 |
| 3.2.4 Stopword Removal | 20 |
| 3.2.5 Tokenization | 20 |
| 3.2.6 Stemming..... | 20 |
| 3.2.6.1 Choosing a Stemmer | 21 |
| 3.2.7 Synonym Matching | 22 |
| 3.2.8 Feature Extraction | 23 |

| | |
|---|-----------|
| 3.2.8.1 Bag-of-Words | 23 |
| 3.2.8.2 N-Grams..... | 24 |
| 3.2.8.3 Word Embeddings | 25 |
| 3.2.9 Sample Preparation..... | 27 |
| 3.3 Classification..... | 29 |
| 3.3.1 Naïve Bayes..... | 29 |
| 3.3.2 Support Vector Machines | 30 |
| 3.3.3 Convolutional Neural Network | 32 |
| 3.3.4 FastText Classifier..... | 34 |
| 4. EVALUATION..... | 37 |
| 4.1.1 Evaluation Metrics..... | 37 |
| 4.1.1.1 Accuracy | 37 |
| 4.1.1.2 Confusion Matrix..... | 38 |
| 4.1.2 Results | 39 |
| 4.1.2.1 Naïve Bayes | 39 |
| 4.1.2.2 Support Vector Machine..... | 42 |
| 4.1.2.3 FastText Classifier | 45 |
| 4.1.2.4 Convolutional Neural Network..... | 47 |
| 4.1.3 Summary..... | 49 |
| 5. CONCLUSION | 51 |
| 5.1 Limitations and Future Work..... | 52 |
| REFERENCES..... | 54 |
| APPENDICES..... | 57 |
| Appendix A..... | 57 |
| A.1 Filtered Word List | 57 |
| A.2 Turkish Stopword List..... | 57 |
| A.3 Synonym List..... | 59 |

LIST OF SYMBOLS

| | |
|-------------|---|
| BoW | : Bag-of-Words |
| CLDR | : (Unicode) Common Locale Data Repository |
| CNN | : Convolutional Neural Network |
| DL | : Deep Learning |
| LSTM | : Long-Short Term Memory |
| ML | : Machine Learning |
| NB | : Naïve Bayes |
| NLP | : Natural Language Processing |
| NLTK | : Natural Language Toolkit |
| ReLU | : Rectified Linear Unit |
| RNN | : Recurrent Neural Network |
| SVM | : Support Vector Machine |
| SVML | : Support Vector Machine - Linear |

LIST OF FIGURES

| | |
|--|-----------|
| Figure 3. 1: Preparation process of tweets | 19 |
| Figure 3. 2: Geometry of word vectors | 26 |
| Figure 3. 3: Classification by SVM, the hyperplane and maximum-margin | 32 |
| Figure 3. 4: 1D convolution with size 3, on the text input..... | 34 |



LIST OF TABLES

| | |
|---|----|
| Table 3. 1: Positive emojis and their labels | 12 |
| Table 3. 2: Negative emojis and their labels | 13 |
| Table 3. 3 Valid and invalid tweet samples | 15 |
| Table 3. 4: Incoherent tweets from dataset | 16 |
| Table 3. 5: Samples from manually labeled data | 16 |
| Table 3. 6: Dataset size | 17 |
| Table 3. 7: Sample tweets from final dataset..... | 18 |
| Table 3. 8: Morphological analysis of a Turkish word..... | 21 |
| Table 3. 9: Comparison of Turkish stemmers..... | 22 |
| Table 3. 10: Sample bag of words features | 24 |
| Table 3. 11: Example of n-grams..... | 24 |
| Table 3. 12: Sample preparation phase for a tweet | 28 |
| Table 4. 1: Accuracy results for emotions, NB classifier..... | 40 |
| Table 4. 2: Accuracy results for groups, NB classifier | 40 |
| Table 4. 3: K-fold accuracy results, for NB classifier with 3-gram features | 41 |
| Table 4. 4: Confusion matrix for emotions, NB classifier | 41 |
| Table 4. 5: Confusion matrix for groups, NB classifier | 41 |
| Table 4. 6: Average precision-recall and f-scores, NB classifier | 42 |
| Table 4. 7: Accuracy results for emotions, SVM classifiers | 42 |
| Table 4. 8: Accuracy results for groups, SVM classifier | 43 |
| Table 4. 9: K-fold accuracy results, SVM classifiers with 3-gram features | 43 |
| Table 4. 10: Confusion matrix for emotions, SVML classifier | 44 |
| Table 4. 11: Confusion matrix for groups, SVML classifier | 44 |
| Table 4. 12: Average precision-recall and f-scores, SVML classifier | 44 |
| Table 4. 13: Accuracy results for emotions, FastText classifier | 45 |
| Table 4. 14: Accuracy results for groups, FastText classifier..... | 45 |
| Table 4. 15: Confusion matrix for emotions, FastText classifier..... | 46 |
| Table 4. 16: Confusion matrix for groups, FastText classifier | 46 |
| Table 4. 17: Average precision-recall and f-scores, FastText classifier | 46 |
| Table 4. 18: Accuracy results for emotions, CNN classifier..... | 47 |

| | |
|---|-----------|
| Table 4. 19: Accuracy results for groups, CNN classifier | 47 |
| Table 4. 20: Confusion matrix for emotions, CNN classifier | 48 |
| Table 4. 21: Confusion matrix for groups, CNN classifier | 48 |
| Table 4. 22: Average precision-recall and f-scores, CNN classifier | 48 |



ABSTRACT

Automatic recognition of feelings in a text is a promising research area which has recently gained more importance with the rapid growth of social media websites, mostly microblogs.

The increasing number of user generated text expands the definition of sentiment analysis where the extraction of emotions from user posts becomes a cutting edge. For that reason, the opinion mining becomes a crucial step for the analysis of social behavior in individuals or groups for the detection of trends.

In current applications, the language of emojis is considered as a common way or an interlingua to express the ideas or intensify feelings. However, there are few studies to reveal its effects on Turkish context for overlapped and separate senses. In this study, emojis have been used as an identifier of the emotions in Turkish texts. The emotion analysis has been performed by Support Vector Machines (SVM), multinomial Naïve Bayes (NB), FastText and Convolutional Neural Network (CNN) using test and train sets derived from Twitter corpus. The preparation and preprocessing of the corpus have been accomplished by generating the classifiers; groups and emotions. The manually labeled tweets have also been added to evaluate the generic function of the classifier. The use of corpus in a generic domain present a promising field where different emotion states have been measured.

RÉSUMÉ

La reconnaissance automatique des sentiments dans un texte est un domaine de recherche prometteur qui a récemment pris plus d'importance avec la croissance rapide des sites de médias sociaux, principalement des microblogs. Le nombre croissant de textes générés par les utilisateurs élargit la définition de l'analyse des sentiments, où l'extraction des émotions à partir des publications des utilisateurs devient un avantage. Par conséquent, les sondages d'opinion deviennent une étape cruciale pour l'analyse du comportement social d'individus ou de groupes en vue de détecter des tendances. Dans les applications actuelles, le langage des emojis est considéré comme un moyen commun ou un interlingua pour exprimer les idées ou intensifier les sentiments. Cependant, peu d'études ont révélé ses effets sur le contexte turc en ce qui concerne les sens superposés et séparés. Dans cette étude, les emojis ont été utilisés comme identifiant des émotions dans les textes turcs. L'analyse des émotions a été effectuée par les machines à vecteurs de support (SVM), le multinomial Naïve Bayes (NB), FastText et le réseau de neurones convolutionnels (CNN) à l'aide de jeux de tests et d'entraînements dérivés du corpus Twitter. La préparation et le prétraitement du corpus ont été réalisés en générant les classificateurs; groupes et émotions. Les tweets étiquetés manuellement ont également été ajoutés pour évaluer la fonction générique du classificateur. L'utilisation de corpus dans un domaine générique constitue un domaine prometteur dans lequel différents états émotionnels ont été mesurés.

ÖZET

Metin içerisinde ifade edilen duyguların otomatik olarak bulunabilmesi, son zamanlarda başta mikro bloglar olmak üzere sosyal medya sitelerinin yükselişi ile önem kazanmış, gelecek vaat eden bir araştırma konusudur. Kullanıcılar tarafından üretilen metin sayısındaki artış; duygu analizinin tanımına farklı bir anlam kazandırmış, kullanıcı mesajlarından duyguları çıkarabilmek ise bu konu içerisindeki öncü yöntem olmuştur. Bu sebeple de fikir madenciliği, trendleri tahmin edebilmek için, grupların veya bireylerin sosyal davranışlarının analizi konusunda kritik bir çalışma konusu haline gelmiştir.

Güncel uygulamalar içerisinde, emoji'ler fikirleri ifade edebilmek veya metin içerisindeki duyguları pekiştirebilmek için diller arası bir araç haline gelmiştir. Bununla birlikte emoji'lerin Türkçe metinlerdeki duyguların ayrıştırılabilmesi veya kesişmesi üzerindeki etkisini ölçen çalışma sayısı çok az olmuştur. Bu çalışmada, emoji'ler Türkçe metinler içerisindeki duyguları tanımlayabilme amacıyla kullanılmıştır. Duygu analizi, Destek Vektör Makineleri (DVK), kategorik Naïve Bayes (NB), FastText, Evrişimli Sinir Ağı (ESA) sınıflandırıcıları ile, eğitim ve test kümeleri Twitter mesajlarından elde edilen veri kümesi üzerinde gerçekleştirilmiştir. Veri kümesinin hazırlık ve işleme aşaması "duygu" ve "grup" isimli iki sınıflandırma şekli dikkate alınarak yapılmıştır. Ayrıca, jenerik sınıflandırma performansını ölçebilme amacıyla elle etiketlenen tweet'ler de değerlendirme işlemlerine dahil edilmiştir. Bağlam barındırmayan bir veri kümesinin kullanımı, farklı duyguların ölçülebildiği bir çalışma alanı olmaktadır.

1. INTRODUCTION

As people become more involved, the computers have gone far beyond a computing device and have begun to serve many different purposes. For most users, they have become a means of communication and self-expression and used to serve people's psychological and emotional needs.

In the first years following the emergence of the World Wide Web (WWW), many web pages have greatly increased the communication capabilities provided by the Internet by providing static information. In the following years, both the development of technology and the widespread adoption of the web started the process called Web 2.0; ordinary users who are not able to create their own web sites started to consume and produce content on the web. Users began writing blog posts, uploading videos, and sharing their reactions and emotions with other people. "Social media" term became the name of all of these platforms where people produced content in a mass manner.

People use several communication forms for both written and spoken languages to express their feelings with their facial expressions, mimics and gesture. In online communication, they use the words and structures from the written texts, but they use different modalities to express or intensify feelings, like smileys, emojis, or memes.

In recent years, the emoji has become one of the fastest and most suitable ways to present a feeling. It has started to become a new modality like sign language depending on the context where the source language is linked to the emotions. The first emojis were a group of 176 icons that helped customers of a company to interact in a better way, created by Shigetaka Kurita, an engineer at a Japanese phone company, in 1998. Now, more than 1,800 emojis exist; and they are defined by Unicode, since version 6.0. Each emoji has

its own Unicode key, which is recognized by the software platform that emoji is being used. Software platforms like social media sites, operating systems, mobile applications have their own implementation of emojis in a standardized way.

Since users have been generating more data from the emergence of social media sites, identifying and classifying the user generated texts has become an important field and created many different sub fields that serve many different purposes. Extracting sentiment from texts is considered one of these tasks, which is called sentiment analysis or emotion analysis, helped many people or companies to analyze what people think about them and which way the feelings are directed towards; positive or negative. Usually the human generated texts are being gathered through web, they are labeled by humans manually, and they are used to create a model which is going to help classifying new texts.

In most of the sentiment analysis studies, the collected data have a clear context, it usually consists of user reactions about a product or a service, such as movie reviews. On the other hand, there is lack of a generic analysis that cover domain independent data; for example, human dialogues, daily expressions, reactions to current events. The social media can be considered as a stream for the data with such characteristics. In this study, in order to label the feelings in such a large corpus, emojis have been used, which has become an interlingua for expressing the feelings. In this study, it is also respected to keep the tweets with no clear context to see its effects on evaluation. Thus, the dataset used in this study contains mostly daily expressions or conversations.

Emoji based sentiment analysis is being studied in social media for different languages. Turkish language bears its own challenges because of its structure, and some studies have proposed ways to overcome this language depending challenge. A mainstream approach, which is also followed in previous studies for Turkish natural language processing (NLP) is chosen in this study; but to it is proposed noted that considering the agglutinative nature of Turkish language and lack of emoji models, there is still a need for better classifiers to be found.

1.1 Aim and Guidance

In this study, the effects of a general corpus have been investigated for the sentiment analysis problem on Turkish texts. Thus, a sentiment analysis has been made, on domain-independent Turkish social media data. Initially, a Turkish corpus formed by tweets has been created, using Twitter API. For the purpose of sentiment analysis, emojis have been used as labels in supervised learning; a list of emojis have been selected to label emotions in texts. These emojis have been grouped with four emotion states; happiness, love, sadness and anger. Then, these emotions have also been divided to two groups; positive or negative to identify if the emotion is a positive feeling or not. A relatively small portion of tweets has been extracted and isolated from the training and test dataset and they have been labeled by human annotators, manually. These manually labeled data has been used to evaluate the classifier performance alongside the test dataset.

After gathering the data, several natural language processing steps have been applied on social media texts, respecting the Turkish language structures and grammar rules.

The classification has been implemented using Support Vector Machines (SVM), multinomial Naive Bayes (NB), FastText classifier and Convolutional Neural Network (CNN) on both sets; emotions and groups. The results have been gathered from all classifiers and evaluated in metrics such as accuracy, precision, recall and f-score. After that, the scores on the effects of emojis for a generic Turkish corpus have been presented. Consequently, the evaluation has been concluded by revealing best scores in a comparative analysis and presenting prospective steps.

The contents of the following sections are described below.

In chapter two, the literature survey can be found. This chapter gives a brief about the sentiment analysis studies so far, including the studies on Turkish data and studies on English data based on the emoji. In chapter three, the methods are explained. The creation and preparation processes of the data are stated, then the chosen classification methods

are explained. The reasons of choosing these classifiers and their working procedures are clarified. In chapter four, the results of classifiers are presented, and comparisons are performed. In chapter five, the outcome of the study and its conclusions are mentioned. Positive and negative aspects of this study are stated, compared to the existing studies so far. The last chapter is consisted of the references.



2. LITERATURE REVIEW

Emotion or sentiment analysis has drawn attention for years, even before the rise of social media. For this study, sentiment analysis studies in Turkish have been explored, and methods of those studies have been discussed. Then, Turkish natural language processing studies have been explored, in order to reveal challenges of Turkish language in this field. Additionally, related studies on emojis have been studied.

2.1 Sentiment Analysis Studies on Turkish Texts

Sentiment analysis on Turkish texts may be grouped into three categories according to the methods utilized in these studies. There are ML-based studies, which mostly depend on supervised learning; then, there are lexicon-based studies, in which sentiments are classified by polarity lexicons which contains Turkish words and their polarity scores. And there are hybrid approaches, which utilize both machine learning and sentiment lexicons in order to achieve optimal results. In some of the previous studies, polarity lexicons have been developed for Turkish language, and they are available to use in order to support ML-based approaches, thus removing the need of creating a lexicon from scratch.

As a lexicon-based study, Dehkharghani et al. prepared a polarity lexicon from Turkish words. According to this study, a lexicon called SentiTurkNet was prepared with words which are important in expressing emotions. In this lexicon a positive, negative or neutral score was calculated for each word. These lexica were tested on Turkish movie reviews dataset. The accuracy score was considered to be low (66.7%). As a result, the lexicon studies were evaluated as weak in capturing indirect expressions and the general meaning of the sentence.

Vural et al. did lexicon-based research on Turkish movie reviews. In order to do this, the sentiment analysis library named SentiStrength was first translated into Turkish and then scored with this library. It has been determined that the results were almost equal to the supervised methods.

Machine learning based studies have produced more successful results than lexicon-based studies. These studies were mostly implemented with supervised learning processes; texts were turned into feature vectors; supervised classifiers were trained, and outcome of the classifiers were presented. However, most of them made measurements on a restricted set of data, which were also domain dependent.

In the work by Türkmenoğlu Twitter and film commentaries were used as dataset. SVM, Naive Bayes and Decision Trees were used as classification methods. Negative words were handled specially during pre-processing. As a result, the classifiers on the set of film commentaries had a higher hit.

Meral et al. studied on the Twitter dataset, they tried n-grams and word-based approaches on domain dependent and independent data sets. As a result, n-gram experiments were found to be more accurate.

Boynukalın made an emotion analysis similar to the one in this study. She classified the texts into four emotions; anger, fear, shame, happiness. The data were formed in two different sets, one set consisted of sentences from a fairy tale with corresponding emotions to those sentences and other set was ISEAR psychological survey asking which emotions they feel in specific situation. She tried Naive Bayes, Complement Naive Bayes and Support Vector Machine classifiers and Complement Naive Bayes was found to be the most successful among them, with approximately 80 percent accuracy.

Kaya et al. did sentiment analysis of Turkish political news in social media. Four different classifiers were used, with character-based n-grams. In contrast to common studies, they concluded Maximum Entropy and the n-gram were more accurate than the SVM and Naive Bayes.

Gezici and Yanıkoglu proposed a complete approach, they combined supervised learning and lexicon-based approaches, making use of a Turkish polarity lexicon called SentiTurkNet. They also used a list of seed words that reinforces the analysis, which they thought helpful to a generic lexicon-based approach. In the end, they achieved 75% accuracy on binary classification of movie reviews in Turkish. They proposed that; for domain dependent data, specialized lexicons should be built, and even a small set of domain-dependent seed words with a large domain independent polarity lexicon have a recognizable effect on the classifier accuracy.

In order to support further studies, some studies tried to detect challenges that may arise in the NLP operations of the Turkish language and proposed solutions to those problems.

Oflazer and Saraçlar presented interesting and challenging features of Turkish language for natural language and speech processing. They explained general outline of Turkish morphology, as it is an agglutinative language with morphemes attaching to a root word. Then they explained the constituent order; how words are placed in sentences and to what extent they are placed in an alternative order. In the end they presented the solutions to these challenges from studies done by that time, and they presented the state-of-the-art tools and applications which are helpful to do more research.

Mulki et al. investigated the effects of preprocessing methods on sentiment classification. The dataset was of Turkish movies and products reviews. Several combinations of preprocessing techniques were applied, and both supervised learning and lexicon-based classifiers were trained. Combinations of the same dataset were used to train classifiers (stopwords removed, stemmed, negation applied etc.). In the end lexicon-based classification resulted in a poor performance, and supervised classifiers resulted when stopwords were removed and negation was applied.

Sak et al. listed a set of resources and tools for exploiting Turkish morphology in natural language processing applications. They provided a morphological parser, a disambiguator and a corpus that includes approximately 500 million tokens, gathered from the web.

Yıldırım et al. investigated and measured the effect of the NLP steps on the sentiment analysis of Turkish social media texts. The experiments showed that the performance improved by %5 percentage points.

2.2 Emoji Based Studies

Emoji-based studies focus on revealing the use cases of emojis and the meanings that they explicitly or secretly add to the texts they are used in. In these studies emojis are treated as words in a sentence, therefore as word lexicons and word embeddings, emoji lexicons and embeddings have been created and analyzed.

Novak et al. provided first emoji sentiment lexicon, called the Emoji Sentiment Ranking. The lexicon contained the sentiment content of 751 most frequently used emojis. The sentiment map of the emojis are created from the sentiments of the tweets they belong to, and emotions of those tweets were labeled by humans.

Shiha et al. analyzed social media data, which were about some positive and negative real-life events, to find out if there was a correlation between emoji usage and events. They did sentiment analysis with and without considering the emoji in sentences and observed that considering emoji in sentiment analysis increased sentiment scores. While Emoji characters are used for expressing both negative and positive opinions, the usage of Emoji characters seemed to intensify the positive opinions more, compared to the negative events.

Li et al. proposed methods to discriminate different kinds of emoji, especially for those with similar meanings. They thought this problem was going to make sentiment analysis harder since the emoji is being used in sentiment analysis, thus a neural network architecture to learn emoji embeddings jointly in order to implement classification. A convolutional neural network (CNN) was trained to get the embedding of human statements and match it to the emojis.

Eisner et al. proposed that there had been many different pre-trained word-embedding vectors sets, but they did not contain any emoji representation. So, they released

emoji2vec, which is a pre-trained embedding for emojis defined by Unicode, and their descriptions were learned by the Unicode standards. The outcome was ready to be used with word-embeddings in any NLP task. In the case of sentiment analysis, they showed that the emoji embeddings outperformed the skip-gram model trained on a large collection of tweets while avoiding the need for contexts.



3. METHODOLOGY

In this section, implementation techniques of the analysis are clarified. The data collection and labeling processes, data preparation steps and the classifiers that have been chosen for the analysis are explained in detail. These steps can be briefly explained as follows:

- Data gathering is the first step, which is aimed to collect the needed data for the analysis. This phase involves receiving tweets from Twitter Stream API by providing the emojis that have been defined and labeling the tweets according to these emojis.
- Preparation consists of several steps of transforming texts into set of features. These steps include text-related operations like stop-word removal and stemming which are standard in most natural language processing applications. Furthermore, feature vectors are created to train classifiers.
- Classification section contains the description of supervised classifiers that are used to train the data in order to create a model to fit new data. In the end, a classifier will be created that predicts the classes of new tweets correctly.

3.1. The Corpus

The dataset consists of tweets shared by users on Twitter. Tweets are multimodal messages that can contain up to 280 characters of text, but can also contain various hypermedia components (images, videos, links) or *hashtags*, which allow messages on Twitter to be tagged on certain topics (e.g. *#tbt*) or *mentions* that make Twitter users message to each other (e.g. *@jack*). Users can post their own tweets, but also share tweets already posted by other users (*retweets*).

There are various ways to collect data from Twitter. The selection of the appropriate method for the analysis is a very important task. A large number of tweets were already posted in the past, and a large number of tweets are also being shared currently. A big portion of tweet data was not relevant for this study. Firstly, because there were a number of misleading contents, such as bot messages and advertising messages. In addition to this, since the aim of this study is to create a generic text classifier, the media components such as links, hashtags or retweets were ignored, because tweets containing hypermedia are meaningful in their own context, so it would require an extra workload to understand what actually being meant in the tweet.

Retweets, mentions and tweets with hashtags were also discarded. Repeated tweets were avoided by filtering out the retweets. Tweets with mentions are ignored, because they probably include users talking about each other's previous tweets. For that reason, the tweet set was designed to include tweets with textual information and emoji properties.

3.1.2 Definition of the Emoji Set

In this study, since the emoji is used as a label in a supervised learning mechanism, selecting the right emojis and matching them with the right groups has great importance.

Unicode Consortium¹ provides a complete list of current emojis² in social media as they are enlisted in a dictionary. Each emoji has been defined with a code and a unique name in CLDR (Unicode Common Locale Data Repository). Emoji names provide a better tracking besides their codes in classification for sentiment analysis. Several emojis have been selected from this list manually, and each emoji has been matched its real-life emotion, then its group; which means positive or negative.














Firstly, emojis with positive feelings have been grouped. These were the emojis that show positive facial expressions; like smile, laugh or hug. The list of potential positive emojis

¹ <https://unicode.org>

² <https://www.unicode.org/emoji/charts/full-emoji-list.html>

has been selected according to the meaningful high frequency rate during the creation of corpus. As the list of emojis grows with new applications, emojis with high frequencies have been kept during preprocessing. The candidates for positive emotions have been found as happiness, joy and love. The corresponding positive emojis has been enlisted in Table 3.1.

Table 3. 1: Positive emojis and their labels

| Emoji | CLDR Name | Emotion | Group |
|---|---------------------------------|-----------|----------|
|  | Grinning | Happiness | Positive |
|  | Grinning Face with Big Eyes | Happiness | Positive |
|  | Smiling Face with Smiling Eyes | Happiness | Positive |
|  | Slightly Smiling Face | Happiness | Positive |
|  | Grinning Face with Smiling Eyes | Joy | Positive |
|  | Face with Tears of Joy | Joy | Positive |
|  | Grinning Squinting Face | Joy | Positive |
|  | Smiling Face with Heart Eyes | Love | Positive |
|  | Kissing Face | Love | Positive |
|  | Face Blowing a Kiss | Love | Positive |
|  | Kissing Face with Smiling Eyes | Love | Positive |
|  | Hugging Face | Love | Positive |
|  | Kissing Face with Closed Eyes | Love | Positive |

Furthermore, negative emotion states have been gathered in a similar fashion. The emotions to be considered as negative have been defined as anger, sadness, shocked, worry and shame. It was not possible to find meaningful and widely accepted emoji set for shame, so it is removed from the emotion set.

Table 3. 2: Negative emojis and their labels

| Emoji | CLDR Name | Emotion | Group |
|---|----------------------------|---------|----------|
|  | Confused Face | Worried | Negative |
|  | Worried Face | Worried | Negative |
|  | Frowning Face | Worried | Negative |
|  | Slightly Frowning Face | Worried | Negative |
|  | Face with Rolling Eyes | Worried | Negative |
|  | Angry Face | Anger | Negative |
|  | Pouting Face | Anger | Negative |
|  | Face with Symbols on Mouth | Anger | Negative |
|  | Face with Open Mouth | Shocked | Negative |
|  | Hushed Face | Shocked | Negative |
|  | Astonished Face | Shocked | Negative |
|  | Anguished Face | Shocked | Negative |
|  | Flushed Face | Shocked | Negative |
|  | Sad but Relieved Face | Sadness | Negative |
|  | Crying Face | Sadness | Negative |
|  | Loudly Crying Face | Sadness | Negative |



3.1.3 Gathering the Data

Twitter provides a streaming API which makes it possible to track keywords, hashtags or media in real time. If an application is registered, Streaming API sends tweets to the registered application, while the application runs. Stream rules can be defined while registering to the stream, so certain filters can be applied on them. Thus, the API makes it possible to ignore retweets, tweets with hashtags or tweets with hypermedia and also track tweets in a certain language.

In addition to the rules above, a filtered word set is defined, which is helpful to filter out tweets mostly related to current events. The set contains words like ("fenerbahçe", "galatasaray", "maç", ...) and also invalid characters, which may belong to hypermedia ("@", "http", ...). If a received tweet contains any of these words case-insensitively, the tweet is discarded, therefore not persisted for analysis. Filtered words are listed in Appendix A.1.

Another important filtering criteria while gathering tweets is filtering out the tweets that contain emojis belong to different emotion groups. In social media there are many tweets containing contrasting emojis; such as happy and angry emojis being in the same sentence. These tweets spoil the integrity of the dataset; hence they are being filtered out in advance. Example of valid and invalid tweets are presented in table 3.3:

Table 3. 3 Valid and invalid tweet samples

| Original Tweet | Validity | Reason |
|---|----------|---------------------------|
| Piyasalar iyice durgunlaştı. 😞 | Valid | - |
| Üstüne bişey demiyorum 😡 https://t.co/4OCMJdjWKN | Invalid | Contains media |
| İmamoğlu başkanımızdır ! Canımızdır ! | Invalid | Contains filtered word |
| Tuhaf işler peşindeyim 😏😏 | Invalid | Different group of emojis |

In order to track relevant tweets for sentiment analysis, code points that belong to emojis that defined in section 3.1.2 are provided to the API. A developer account has been created to register to the Streaming API and Unicode representations of emojis have been provided as tracking keywords. In a couple of months, approximately 100K tweets were collected.

3.1.4 Review of Dataset

We have gathered a large number of tweets; but before recording classification results, it is necessary to run the preprocessing and classification steps in order to understand cohesion of the data. This is a recurrent process and involves the steps from next chapters. Some parts of the data are found to be cohesive and are going to stay, and some parts are going to be discarded from the final classification.

3.1.4.1 Removing Incoherent Data

After calculating the classifier accuracies with combinations of the data, it has been possible to find which part of the data is relevant and which parts should be discarded. Unfortunately, the data labeled as “*shocked*”, “*worried*” and “*joy*” are found to be lowering the accuracy. The data labeled as “*joy*” is hard to distinguish from “*happiness*” and people have seemed to be using “*joy*” emojis for multiple purposes. Data labeled as “*worried*” is also very similar to “*sadness*”. “*Shocked*” dataset has seemed to be lowering the groups’ accuracy, because the tweets with surprised or shocked emotions do not always represent negative feelings.

Table 3. 4: Incoherent tweets from dataset

| Emotion / Group | Original Tweet | English Translation |
|--------------------|---|--|
| Joy / Positive | Lütfen insafa gelin 😂 | Please have mercy 😂 |
| Worried / Negative | Yarın 20lik dişi çekecekler. Korkudan uyuyamıyorum 😞 | They will remove my wisdom tooth tomorrow. I can't sleep with fear 😞 |
| Sadness / Negative | Allahım ne olur dursun korkudan uyuyamıyorum 😞 | My God, please, make it stop, I can't sleep with fear 😞 |
| Shocked / Negative | Bu ne mükemmel bir maç 😱 | What a perfect game is this 😱 |

As it is seen in Table 3.4, first tweet labeled as “joy” is actually a sarcasm, which is impossible to distinguish only by looking at the text. And tweets labeled as “worried” and “sadness” have very similar meanings. And as it can be seen in the last example, “shocked” tweets can be positive, emoji cannot tell if the text is positive or nor.

3.1.5 Manually Labeled Data

In order to prove that our methods output a domain-independent classifier, the classifiers must be evaluated via dataset that was labeled by human annotators.

The manually labeled data has normalized text only; the tweets are stripped off their emojis, to prevent labelers being affected. In the end, 2K tweets have been labeled manually, and 1.2K of them have been used for evaluation.

Sample manually labeled data, with their original texts, normalized texts, emoji labels and human-put labels are shown in table 3.4. Most of the labels given by humans actually coincides with emoji labels, the samples in the table represent the conflicted ones.

Table 3. 5: Samples from manually labeled data

| Original Tweet | Normalized Text | Emoji Label | Manual Label |
|---|---|-------------|--------------|
| Kendini çok özlettiğin için senden nefret ediyorum 😡 | kendini çok özlettiğin için senden nefret ediyorum | Anger | Love |
| Bıraksalar akademik kariyer yapacağım ama bırakmıyorlar 😊 | bıraksalar akademik kariyer yapacağım ama bırakmıyorlar | Happiness | Sadness |
| Canlı gördüm resimdekinden bin kat güzel 😭 | canlı gördüm resimdekinden bin kat güzel | Sadness | Love |
| Yarın mont almak için mükemmel bir gün olabilirrrr 😍 | yarın mont almak için mükemmel bir gün olabilir | Love | Happiness |

3.1.6 Final Dataset

After removing the incoherent data, it is now possible to present a list of sample tweets from the final dataset. The final dataset has four emotions: happiness, anger, love and sadness; with two groups, positive and negative.

40K tweets are selected for testing and training. Apart from them, 1200 tweets are manually labeled and evaluated alongside test and training data. To utilize a better classification, tweet counts are fixed for each emotion and group.

Table 3. 6: Dataset size

| Emotion | Group | Training Data | Testing Data | Manually Labeled |
|-----------|----------|---------------|--------------|------------------|
| Happiness | Positive | 8000 | 2000 | 300 |
| Love | | 8000 | 2000 | 300 |
| Anger | Negative | 8000 | 2000 | 300 |
| Sadness | | 8000 | 2000 | 300 |
| Total | | 32K | 8K | 1.2K |

3.1.6.1 Sample Tweets

Sample tweets from tweets can be found in table 3.7. These tweets are completely valid and took place inside either training or test dataset.

Table 3. 7: Sample tweets from final dataset

| Emotion / Group | Original Tweet | English Translation |
|--------------------|---|---|
| Happy / Positive | Aşk rengi benimki 😊 | My color is love 😊 |
| Happy / Positive | Buz patenine merak saldım. Finaller bitsin ilk işim olacak ama. 😊 | I am interested in ice skating. It's my first thing to do after the finals 😊 |
| Love / Positive | Kalbimi burada bırakıyorum 😍 | I leave my heart here 😍 |
| Love / Positive | TRT belgeseldeki filler sizi çok sevdim. Öpüyorum 😘 | The elephants in TRT documentary, I loved you. I kiss you 😘 |
| Anger / Negative | Oksijen israfı 😡 | Waste of oxygen 😡 |
| Anger / Negative | Maşa yaparken yanmadığım tek bir gün bile yok, yaktım yine her yerimi 😡 | There is not a single day that I do not burn myself while using curling iron. I've burnt myself all again 😡 |
| Sadness / Negative | Nolucu benim bu boğazım 😞 | What is going to happen with my throat 😞 |
| Sadness / Negative | Öldüm ben 😭😭😭😭 | I am dead 😭😭😭😭 |

3.2 Preparation

In the preparation phase, tweets are normalized by applying cleaning, lowercasing and word correction methods. Then, stop word removal is applied to remove irrelevant words in the texts. After that, sentences are transformed into token sets in the tokenization phase. When tokenization is done, stemming is applied to change words with their stems.

Stemming is an important step that must be done carefully because of Turkish language structure, explained in more detail in section 3.2.6. Stemming is followed by synonym grouping to reduce the size of vocabulary. In the feature extraction phase, tweet texts are transformed into features. Figure 3.2 gives a brief representation of the whole preparation process.

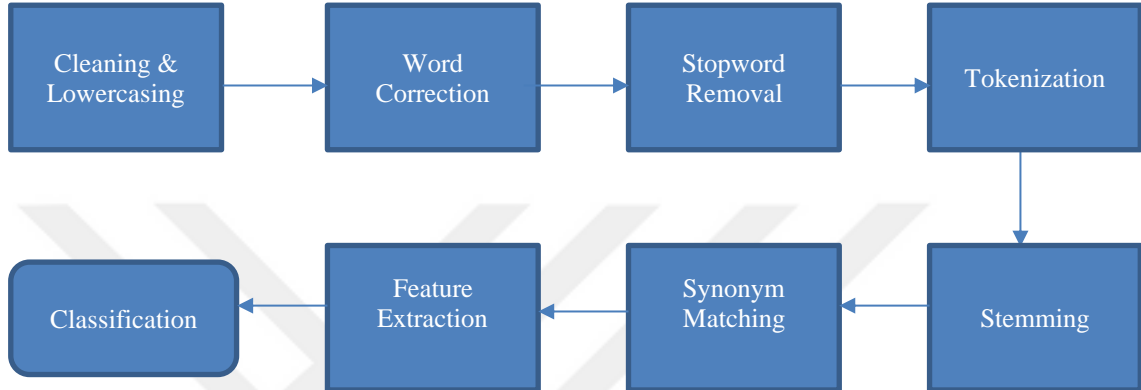


Figure 3. 1: Preparation process of tweets

The phases mentioned here are briefly described in the following sections. Sample tweets are presented with their transformations in all preparation steps in section 3.2.9.

3.2.1 Cleaning and Lowercasing

As the first step, dataset is cleaned from non-alphabetic characters. These include parentheses, punctuation marks, numbers as well as emojis. Emojis have been used to label the data, thus they are not needed in texts anymore. After cleaning, all words in the tweets transformed to lowercase forms.

3.2.2 Word Correction

Our dataset is consisted of user-generated texts, so there are many spelling mistakes. Some of these mistakes are not intentional and some of them are intentional. The reason for intentional errors is that users post tweets to fit social media spelling jargon to attract more attention to their messages. At the same time, messages produced by users who do

not use the Turkish keyboard result in the use of the nearest letter in the English alphabet instead of some Turkish letters. All these errors are corrected with the word correction function of Zemberek library.

3.2.4 Stopword Removal

Stop words are the common words which do not provide any information to a text classification problem. Removing these words increase the performance of classifiers, since the vocabulary is reduced to fewer words.

Turkish WikiDictionary, *VikiSözlük*³ defines the most common words in Turkish. The selection of stopwords and the corresponding removal procedure have been performed through VikiSözlük. The whole list of stop words can be found in Appendix A.2.

3.2.5 Tokenization

Tokenization is the process of transforming the texts into a set of words; splitting sentences by whitespaces, commas or dots. Tweet texts have been converted to tokens by applying tokenization, which is also provided by Zemberek library as a function.

3.2.6 Stemming

As Turkish is an agglutinative language, new words are mostly derived from Turkish or non-Turkish root words. This makes the processing of the language more complex than any other non-agglutinative languages. In addition, affixes that have been placed for same purposes would be much different in the text; Turkish language has sound harmonies for vowels and consonants, the affixes alter according to the last letters of the words they are integrated to.

Stemming is the process of finding a word's root stem. In the case of Turkish, since there are several suffix types that can follow a stem; there is a trade-off between finding the simplest stem and finding the most meaningful stem. Aggressive stemmers may find the

³ <https://tr.wiktionary.org/wiki/Anasayfa>

simplest stem, which groups most of the words and reduces the vocabulary size; but this may cause loss of meaning in words, therefore reducing the classifier performance.

As an example, "*sorgulandım*" is a Turkish word; which is also a sentence wearing derivational and inflectional suffixes on it. English translation of the word is "*I was questioned*". Morphological structure of the word starts with a verb root and meaning changes with the attached suffixes. In table 3.8 a morphological analysis of the word is presented.

Table 3. 8: Morphological analysis of a Turkish word

| Word | Last Suffix | Suffix Type | Current Meaning |
|-------------|-------------|---------------------------|-----------------|
| sor | - | - | to ask |
| sorgu | - gu | Derivational | query |
| sorgula | - la | Derivational | to query |
| sorgulan | - n | Derivational (Passive) | to be queried |
| sorgulandı | - dı | Inflectional (Past Tense) | was queried |
| sorgulandım | - m | Inflectional (Subject) | i was queried |

In this case, best practice would be to keep "*sorgulan*" as the stem, which is the passive verb root. The suffixes after that point emphasizes the tense and subject, which are not completely changing the meaning of the word.

So, a good idea for stemming would be removing the inflectional suffixes but keeping the derivational suffixes; because the derivational suffixes alter meanings of a word resulting in changing the meaning of the sentence. In addition to this, a stemmer must distinguish carefully if a word actually wears suffixes or not; for example, *yardım* ("help") may look similar with the word *sorgulandım* to a stemmer, because of ending with *-dım*, but in this case the word has no suffixes thus should not be stemmed.

3.2.6.1 Choosing a Stemmer

There are a few stemmer implementations for Turkish language. One of them is developed as a function inside Zemberek framework (Akın & Akın, 2007), as well as other functions provided by the library. Another one is Turkish stemmer built in Snowball stemmer, created by Osman Tunçelli and followed the rules set in the paper by Eryiğit and Adalı, 2004⁴. The last one is Resha stemmer, created by Harun Reşit Zafer. It was developed to be a non-aggressive and fast stemmer, built on a stem dictionary containing 1.1. million word-stem pairs.

In order to pick a stemmer, these implementations have been tried and compared to each other in the following Table 3.9.

Table 3. 9: Comparison of Turkish stemmers

| Word | Snowball | Zemberek | Resha |
|---------------|----------|------------|----------|
| bile | bil | bile | bile |
| kitapçık | kitapçık | kitapçık | kitapçık |
| yardım | yardım | yardım | yardım |
| sorgulandım | sorgulan | sorgulan | sorgulan |
| gideceğim | git | gidecek | gidecek |
| istanbullular | istanbul | istanbullu | istanbul |

The outcome of the stemmers that are tested with the same words are very similar, while Snowball is a bit more aggressive compared to the others. According to these results, as it is already being used in this study, Zemberek has been chosen as the stemmer for Turkish tweets.

3.2.7 Synonym Matching

⁴ <https://github.com/otuncelli/turkish-stemmer-python>

⁵ <https://github.com/hrzafer/resha-turkish-stemmer>

Finally, as the last operation, synonyms words are grouped into a single one. The grouping of synonym words also reduces the vocabulary size thus increasing the classifier performance.

We have defined a synonym list in Appendix A.3. According to this list, all synonym groups are read, and first word is selected as the root synonym of the group. If any other word other than the first word is encountered in the text, it is replaced by the first word in the group.

3.2.8 Feature Extraction

Feature extraction, or "feature engineering" as a more general term, is the process of transforming attributes of the data into shapes that can be used by machine learning algorithms. A successful feature extraction causes classifiers to output optimal results. To implement a successful feature extraction, the domain characteristics of the data must be known well in advance.

In the context of text classification, the text data must be converted into feature vectors. There are different ways to obtain relevant features from the text for classification problems, and representation of feature vectors also vary in shapes and sizes.

3.2.8.1 Bag-of-Words

Bag-of-words model (BoW), is a very common way of extracting features from text to use in machine learning (ML) algorithms. Using bag-of-word features, one can have a vector where each element of the vector indicates the existence or non-existence of the word from the vocabulary in the sentence. As an alternative, the element can indicate the occurrence count of that word in the document (Goodfellow et al., 2016).

The model is only concerned with whether words occur in the document, or how many times do they occur, not where they occur in the document. This causes losing of the context; grammatical structure of the sentence is ignored, and adjacent words also lose

their integrity. Phrases that are consisted of multiple words become meaningless to the classifiers.

In order to overcome the obstacles of BoW model, n-gram features are used. N-grams are helpful to capture the adjacent words as features thus making the training features more context aware.

Table 3. 10: Sample bag of words features

| State | Result |
|-----------------|--|
| Original | Kalıyorum yavaş yavaş 🙄 Sanırım daha fazla okuyamicaaaammm |
| Stemmed | kal yavaş yavaş san daha fazla okuyamicam |
| BoW - Boolean | {kal:True, yavaş:True, san:True, daha:True, okuyamicam:True} |
| BoW - Numeric | {kal:1, yavaş:1, san:1, daha:1, okuyamicam:1} |
| BoW - Occurence | {kal:1, yavaş:2, san:1, daha:1, okuyamicam:1} |

3.2.8.2 N-Grams

N-grams are combinations of adjacent tokens or combinations of letters forming the tokens. They are helpful to capture word order in a sentence as well as identifying the entities named by multiple words, such as "New York", or "Elvis Presley". Therefore n-grams are very powerful complements to BoW features. n denotes the taken combinations of tokens in a text.

As an example, "I love text classification" text can be transformed into following word-level n-grams, presented in table 3.11.

Table 3. 11: Example of n-grams

| Features | Result |
|----------|--|
| Original | I love text classification |
| 1-gram | [(I), (love), (text), (classification)] |
| 2-gram | [(I love), (love text), (text classification)] |
| 3-gram | [(I love text), (love text classification)] |

Longer the n-gram is (higher the n), more context can be captured, however the feature set would be larger thus making it harder for classifiers to be trained. Longer n-grams may also cause classifiers to be overfit; meaning classifiers' learning of training data features more than wanted, which decreases the success of the classifier against the test data. Optimal length of n-grams is dependent on the vocabulary that is used, and the domain of the data belongs to.

3.2.8.3 Word Embeddings

Word embeddings are vector representations of words, sentences or documents. They are used to find semantic and syntactic similarities of words in a document or their relations with other words within the same context. It is a method to model natural language in order to implement feature engineering for NLP studies where natural language words, sentences or documents are mapped into vectors of numbers. The complex and high-dimensional space of words are reduced into a vector space with smaller dimensions. Within these low dimensional vector space, math between vectors can be implemented, similarities can be calculated and then the results can easily be converted back to the natural language.

In 2012, by a team of researchers led by Thomas Mikolov, a new way to represent word embeddings was found and published in a research paper (Mikolov et al., 2013). Named as word2vec, the algorithm uses deep learning and neural networks-based techniques to convert words into corresponding vectors in such a way that the semantically similar vectors are close to each other in N-dimensional space, where N refers to the dimensions of the vector.

Word2vec is trained by an unsupervised learning algorithm. Vocabulary is not labeled; the algorithms learn the representations of the words by considering the neighbors of a word in the sentence. Therefore, if a larger corpus is provided, the algorithm learns more representations and relations of the words.

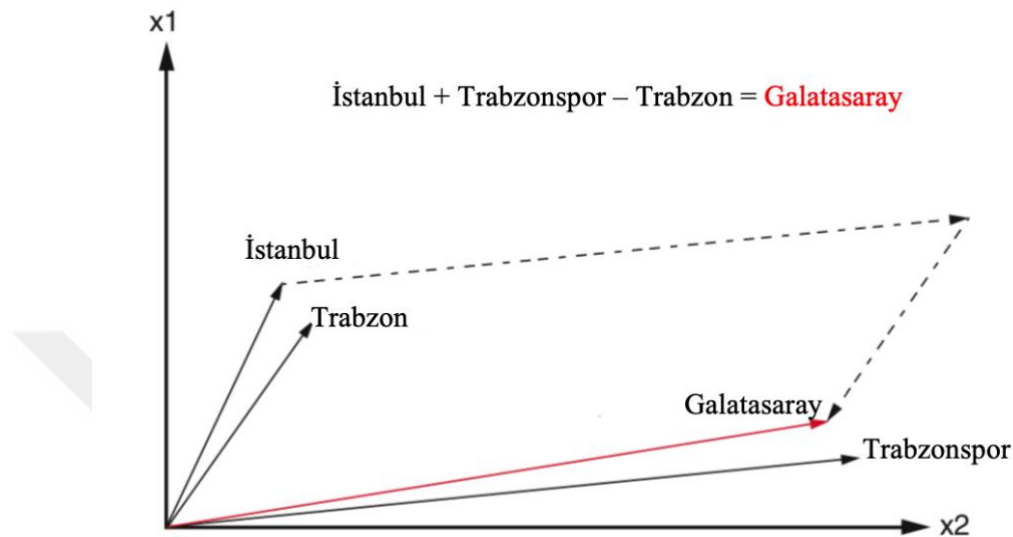


Figure 3. 2: Geometry of word vectors

Once computed, word2vec model contains implicit knowledge about the relationships of words, including similarity. As seen in Figure 3.2 the vectors of two cities; Istanbul and Trabzon are approximately in the same distance to their football teams, Galatasaray and Trabzonspor. Therefore, word vectors can be used to question analogies between words; such as "*Trabzonspor is to Trabzon as what is to Istanbul?*", and the answer would be Galatasaray.

To compute Word2vec representations, there are two main approaches:

- The *skip-gram* approach, which predicts the neighboring words from a single input word.
- *CBOV* approach, predicts the target word from neighboring words.

It is not always mandatory to compute word vectors to make use of them in a study. There are several sets of pretrained vectors for each language, trained on large vocabularies.

Pretrained sets can be downloaded and used for generic tasks; however, if the data belong to a domain that has unique or rare terms; such as medical texts or historical scripts, using generic word embeddings would not be suitable therefore word vectors should be calculated from the dataset itself.

The idea behind word2Vec also expanded to represent sentences and documents, named in practice as sentence2vec and doc2vec. Following the emergence of word2vec, several alternatives have also been discovered. GloVe, Bert and FastText are some of the frameworks that compute word embeddings as an alternative to word2vec.

FastText⁶, developed in Facebook's AI Research (FAIR) lab, is known to be a faster alternative to previous word embedding methods. It allows to create both supervised and unsupervised learning for vector embeddings, which can also be optimized in order to allocate smaller space and provide faster model loading.

FastText also provides pretrained vectors for most languages, including Turkish⁷. In this study pretrained Turkish vectors are also used alongside the model that is trained using the tweet corpus.

In this study, FastText is chosen as a provider for word embeddings. In order to make use of word embeddings, two models are being used; one is pretrained Turkish vector embeddings provided by FastText, and the other is the model that is trained from tweet corpus. The vector features obtained from FastText are used to train convolutional neural network classifier.

3.2.9 Sample Preparation

In table 3.9 preparation phase for a single tweet is presented. Each operation in the preceding sections have been applied to the tweet and operations with their effects on the tweet are explained.

⁶ <https://fasttext.cc/>

⁷ <https://fasttext.cc/docs/en/crawl-vectors.html>

Table 3. 12: Sample preparation phase for a tweet

| Preparation Step | Current State | Operation |
|---|--|---|
| Original tweet | Her yılbaşında inanılmaz kararlar verip bir kaç gün içinde hepsinden caymak ! En sevdiğim ! 😊😊 | None |
| Cleaning | her yılbaşında inanılmaz kararlar verip bir kaç gün içinde hepsinden caymak en sevdiğim | Emojis and exclamation marks cleaned |
| Word Correction | her yılbaşında inanılmaz kararlar verip bir kaç gün içinde hepsinden caymak en sevdiğim | <i>inanılmaz</i> to <i>inanılmaz</i> correction |
| Stopword Removal | yılbaşında inanılmaz kararlar verip gün içinde hepsinden caymak sevdiğim | <i>her, bir, kaç, en</i> stopwords removed |
| Tokenization | [yılbaşında, inanılmaz, kararlar, verip, gün, içinde, hepsinden, caymak, sevdiğim] | Sentence converted into token set |
| Stemming | [yılbaş, inanılmaz, karar, verip, gün, iç, hepsi, caymak, sevdik] | Stemming applied to the remaining words |
| Synonym Matching | [yılbaş, inanılmaz, karar, verip, gün, iç, hepsi, caymak, sevdik] | None, no synonyms found |
| Feature Extraction (Bag-of-Words) | {yılbaş:True, inanılmaz:True, karar:True, verip:True, gün:True, iç:True, hepsi:True, caymak:True, sevdik:True} | Tokens converted to bag of words features |
| Feature Extraction (Bag-of-Words, 2gram) | {(yılbaş, inanılmaz),(inanılmaz, karar),(karar, verip),(verip, gün), (gün, iç), (iç, hepsi),(hepsi, caymak) (caymak, sevdik)} | Tokens converted to 2-gram features |
| Feature Extraction (Bag-of-Words, 3gram) | {(yılbaş, inanılmaz, karar),(inanılmaz, karar, verip),(karar, verip, gün), (verip, gün, iç), (gün, iç, hepsi),(iç, hepsi, caymak),(hepsi, caymak, sevdik)} | Tokens converted to 3-gram features |
| Feature Extraction (Word Vectors) | {([-1,124124, 0,19494, 0,888], ...)} | Word vectors are obtained from FastText |

3.3 Classification

In sentiment detection, the initial model is set on a text classification problem. Actual routines in text classification serve to interpret single or multiclass approaches. In this study, Python language, Natural Language Toolkit (NLTK) alongside scikit-learn, FastText framework and Keras are used to classify the sentiments derived from the corpus. The data has been labeled as emotions and groups. Thus, the classifiers have been assigned to the tasks of classifying groups and emotions. In order to understand the emoticons and their relationship with the feelings in the tweets, Naive Bayes and Support Vector Machines have been used in several steps to repeat the experiments. First, the dataset has been divided into training and test parts. The learning strategy based on learning functions has been set on k-fold cross validation to ensure a robust accuracy in sentiment classification for a generic domain. Both methods have been tested for different training and test sets as emotions and groups. The results have been evaluated with accuracy, precision, recall and F1 scores.

Overall classification and evaluation processes have been implemented in Python language, version 3.7. Versions of NLTK and scikit-learn that have been used is 3.4.4, and 0.21.2 respectively. For CNN classifier, Keras 2.2.4 backed by Tensorflow 1.14.0 has been utilized. FastText framework has been used as a binary application rather than a development kit, and version 0.9.1 has been chosen. As the development environment, PyCharm IDE 2019.2 edition has been used.

3.3.1 Naïve Bayes

In ML, Bayesian classifiers are a group of classifiers, based on Bayes' probability theorem. They predict the probability of a label according to the probabilistic model created during the learning process. When learning is complete, new data are assigned to the class that has the highest probability of containing the new data features.

The Naive Bayesian classifier ignores the conditional probabilities in the data and considers the features as independent of each other (Alpaydın, 2011). Since this is very compatible to the BoW model that is frequently used in NLP applications, Naive Bayesian classification has been successfully applied to text and document classification in many studies, including Turkish texts (Amasyalı & Diri, 2006).

For missing features in the feature set, Naive Bayes classifier needs a smoothing method, which is Expected Likelihood Estimation for Naive Bayes implementation of NLTK framework that has been chosen.

$$P(\text{emoji} \mid \text{features}) = \frac{P(\text{emoji}) \cdot P(\text{features} \mid \text{emoji})}{P(\text{features})} \quad (3.1)$$

Equation (3.1) presents the global perspective, where $P(\text{emoji})$ notes prior probability of the sentiment of the emoji in tweets. Thus, $P(\text{features} \mid \text{emoji})$ shows the probability of a given feature set to be predicted as that emotion. This relation is also called as likelihood. $P(\text{features})$ represents the probability of the feature set to exist in training set. Probability of given features belong to the emotion class represented with $P(\text{emotion} \mid \text{features})$. In the end, class with the highest value of $P(\text{emotion} \mid \text{features})$ is going to be selected by the algorithm (3.2).

$$c^* = \arg \max_{j=1, \dots, m} P(\text{label}_j) \cdot P(\text{features} \mid \text{emoji}_j) \quad (3.2)$$

Consequently, the model would fit suitable in the corpus for short sentences and the language model uses lexical features. Naive Bayes is a scalable, fast and efficient classifier when number of features are high. However, since it is being used with BoW features and all features are considered independent, it loses ordering and dependencies between words.

3.3.2 Support Vector Machines

Support Vector Machine (SVM) is a form of maximum margin methods, belongs to a family of methods generalized by the name "kernel machine", that tries to classify features by figuring out a hyperplane and maximizing the margin separating to find optimal separating hyperplane. It is considered among robust techniques for classification and has been preferred in text classification studies for many years (Jurafsky, Martin, 2008).

The goal of SVM is to find the largest gap that separates the data, that means finding the plane with maximum distance from the training points. The hyperplane divides the data but is also as far as possible from the data points. If the data is separable, there are infinitely many hyperplanes that will separate it.

For a set of training points, (\vec{x}_i, y_i) represents the vector \vec{x}_i and its respective emotion class y_i . It is obvious that the shortest distance between a point and a hyperplane is the normal vector \vec{w} . The unit vector is $\vec{w}/|\vec{w}|$. Therefore, the distance between the boundary is used to calculate the margin ρ using equations in 3.3 and 3.4.

$$r_i = \frac{y_i(\vec{w}^T \vec{x}_i + b)}{|\vec{w}|} \quad (3.3)$$

$$\rho = \frac{2}{|\vec{w}|} \quad (3.4)$$

SVM calculates the hyperplane with the maximum margin using equation in 3.4 by respecting $\forall (\vec{x}_i, y_i), y_i(\vec{w}^T \vec{x}_i + b) \geq 1$.

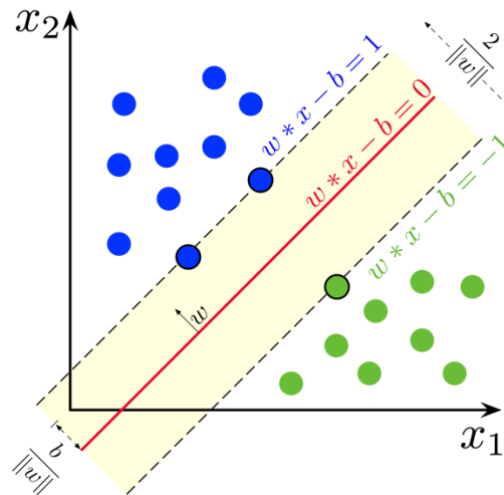


Figure 3. 3: Classification by SVM, the hyperplane and maximum-margin

In Figure 3.3, an SVM is trained with samples from two classes. One class is represented by green dots and other is blue. The hyperplane that divides them have the maximum possible width, so it is found to be the optimal solution. Data points on the margin are named as the support vectors.

There are linear and non-linear approaches for the definition of these hyperplanes. An SVM with linear kernel tries to separate the data by generating a linear function which works best when the dataset is linearly separable. However, when the data is not linearly separable, like a XOR function for a simple example, non-linear kernels are used. In order to separate data; they create non-linear subsets of features to move samples on to a higher-dimensional feature space, where a linear kernel used to separate data in the end.

3.3.3 Convolutional Neural Network

A convolutional neural network (CNN), is a type of neural network architecture, which has gained attention recently because of its use in deep learning applications. CNN is named from the concept of sliding (or *convolving*) a small matrix over the data, thus capturing important features from the data and reducing it to a smaller feature space.

Instead of assigning weights to the features as traditional feedforward neural networks do, a convolutional net defines a set of filters (also known as *kernels*) that slides across the feature set. These filters take snapshots of the data, and outputs of these snapshots

create a new layer of inputs. The distance each convolution moves at each step is known as the "*stride*" and is typically set to 1.

As filters slide over the features, one stride at a time, they take snapshots of the features they are covering at the time. The results of those snapshots are then multiplied by the weight associated with that position in the filter. The products of feature and weight at that position are then summed up, and are moved into an activation function, like rectified linear units (ReLU).

An important term in CNN architecture is the concept of "pooling". Pooling is a way to reduce the size of the representation across layers. Pooling layer is placed between convolutions and is used to select important features for next convolution by reducing the representation, thus increasing the speed of training. Pooling method can be max pooling, which means picking the largest of n numbers in the $n \times n$ area, or average pooling which takes the average of the numbers in the same area.

After convolutions and pooling, fully connected layers take place. They are used to compute class scores that are the output of the classification. The dimensions of the result is $1 \times 1 \times N$ where N denotes the number of labels in training dataset.

In text classification, convolutional network slightly differs. Since the feature set is a sentence, which is a one-dimensional data, the convolution is also a one-dimensional shape, which is called 1-D convolution.

Given a vector of tokens $w_{1:n} = w_1, \dots, w_n$, where each token is associated with an embedding vector of one dimension, A 1-D convolution of width n is the result of convolving a window of size n over the sentence, as seen in figure 3.5.

ben çay sevmiyor değilim sadece başkasının yaptığını sevmiyorum
ben çay sevmiyor değilim sadece başkasının yaptığını sevmiyorum
ben çay sevmiyor değilim sadece başkasının yaptığını sevmiyorum

Figure 3. 4: 1D convolution with size 3, on the text input

A typical CNN architecture for text classification would start with an embedding layer, which takes inputs of word vectors provided by FastText. Dimensions and max sequences in the data (number of words) are provided to create the embedding layer, which is 300 and 40, respectively. Embedding layer is followed by sequential convolutional and pooling layers, which is fed to a dense layer (fully connected NN) in the end.

3.3.4 FastText Classifier

FastText is a library for efficient learning of word representations, and also provides sentence classification, through supervised learning. The idea behind the FastText classification is published in the paper (Joulin et al., 2016) where they explained how they overcame the linear complexity of preceding classifying methods when number of classes or instances are high.

In order to implement text classification, FastText classifier follows the steps below:

- Word representations from sentences/documents are averaged into text representations, and then are fed to a linear classifier algorithm, which is multinomial logistic regression.
- To overcome the complexity of linear classifier, a hierarchical SoftMax layer, based on the Huffman coding tree is used. This reduces the computational complexity $O(kh)$ to $O(h\log(k))$, where k is the number of classes and h is the dimensions of text vector.

- In order to capture word order, simply n-gram features are used. Using n-grams provides efficiency with very minimal losses.
- Prevention of inflation of the n-gram vocabulary is done by "hashing trick"; which maintains fast and memory efficient mapping of the n-grams.

Consequently, FastText trains models significantly than other frameworks. Supervised learning can be customized, altering the following parameters.

- `dim`: Size of word vectors created or used by the classifier. Default value is 100. The size of pretrained Turkish vectors provided by FastText is 300, therefore in order to evaluate classifiers in same conditions, vectors of tweet corpus is going to have the same value.
- `epoch`: Epoch count is the number of times each instance in the data is seen by the classifier. The default value is five.
- `lr`: Learning rate is a common parameter for neural network architectures. It is a measure of how much the model changes after each training step. Larger learning rate causes classifier to take larger steps, therefore it may cause missing the optimal point. On the other hand, smaller learning rate causes slower training.
- `wordNgrams`: As stated in section 3.1, n-grams are adjacent tokens from text. Fasttext classifier can be trained with n-grams, and since the previous classifiers used in this study have been trained by n=1,2,3 n-grams, same options are going to be used
- `pretrainedVectors`: Determines if the classifier is going to use the pretrained word vectors or not. If not provided, the classifier learns the vectors from training data itself.

In this study, two FastText classifiers are used; one is trained with pretrained Turkish word embeddings provided by the framework, one is trained with the tweet corpus. Both classifiers used $n=1,2,3$ grams separately.



4. EVALUATION

In this study, several test procedures have been used to reveal the best classifiers according to experiments. In order to evaluate classifiers, train and test data are separated and classifier performance of test data and manually labeled data are scored. The current parameters of evaluation are accuracy, precision, recall, f1-score and confusion matrixes.

The classifiers are trained with different class set combinations of the same dataset, groups and emotions. These class sets are:

- Emotions: *happiness, love, anger, sadness* (4 classes)
- Groups: *positive, negative* (2 classes)

4.1.1 Evaluation Metrics

4.1.1.1 Accuracy

In most evaluation processes, accuracy is the first measure that is considered. It is simply the percentage of number of correctly predicted instances to the number of all instances. The number of true positives (T_p), true negatives (T_n), false positives (F_p) and false negatives (F_n) have been used in the following expression;

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \quad (3.5)$$

Despite being used commonly for the evaluation of classifiers, accuracy can be misleading in some cases. When classes are imbalanced or data characteristics is not distributed randomly enough, the classifiers may result too high or low accuracy. As a

result, in order to complement accuracy results, confusion matrix should be used and measures such as precision and recall should be calculated.

4.1.1.2 Confusion Matrix

A confusion matrix presents the results of a classifier; it contains actual and predicted labels of the classified instances by a classification algorithm. Performance of the classifiers has been evaluated by calculating measures such as precision and recall.

In the following sections, confusion matrixes for some of the classification experiments are provided. The columns represent the predicted classes, and rows represent the actual classes. Therefore, the main diagonal of the matrixes states true positives (highlighted in bold) where actual class values and predicted class values have been found similar.

Precision is a measure of relevancy. It is calculated as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p) as follows;

$$Precision = \frac{T_P}{T_p + F_p} \quad (3.6)$$

Recall is a score for measuring what percentage of the positives were identified correctly by the classifier. It is calculated as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n) as follows;

$$Recall = \frac{T_P}{T_p + F_n} \quad (3.7)$$

When precision is high, the classifier may return very few results, but most of them are true positives. On the other hand, if recall is high, more results are returned but many of them may be incorrect. As a result, there is a trade-off between precision and recall; higher

precision means a low false positive rate, and higher recall means a low false negative rate.

An ideal classifier has high precision and high recall, returns many results, and all results labeled correctly. As a simpler metric, F1 score, is just the harmonic mean of precision and recall, consequently higher F1 score stands for better performance at any case.

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (3.8)$$

Precision, recall and f-scores are calculated as micro and macro averaged. Macro-averaged metrics are first calculated individually for each class, and then their average is taken, hence treating all classes equally even if they differ in data size. Micro-averaged values are aggregated results of all true positives, false negatives and false positives. Macro averaged results converge to the class with largest samples, while micro averages converge to the smallest ones. Since the dataset of this study is balanced across all classes, classes are not weighted, micro-averaged results are going to be equal.

4.1.2 Results

Evaluation results have been gathered upon the completion of the classifiers' training, where training, test and manually labeled data have been fit to the trained models separately. The classifiers provided by NLTK (NB and SVM) have had the built-in support for evaluation; however, for FastText and CNN, manual gathering and calculation of the results have been required.

4.1.2.1 Naïve Bayes

In the accuracy results of Naive Bayes (NB) classifier for emotions in Table 4.1, it is seen that highest accuracy results for the classification of the emotions have been obtained in

3-gram features. With 3-grams training accuracy is found to be 0.97 and test accuracy is 0.65. Manually labeled set has very close scores to the test set, being 0.65 maximum.

Table 4. 1: Accuracy results for emotions, NB classifier

| Features | Training Set | Test Set | Manually Labeled Set |
|----------|--------------|----------|----------------------|
| 1-gram | 0.86 | 0.61 | 0.62 |
| 2-gram | 0.95 | 0.63 | 0.63 |
| 3-gram | 0.97 | 0.65 | 0.63 |

For groups, higher results are observed. The classifier has also better results for manually labeled data this time as presented in the table below.

Table 4. 2: Accuracy results for groups, NB classifier

| Features | Training Set | Test Set | Manually Labeled Set |
|----------|--------------|----------|----------------------|
| 1-gram | 0.87 | 0.74 | 0.74 |
| 2-gram | 0.98 | 0.77 | 0.79 |
| 3-gram | 0.98 | 0.78 | 0.79 |

In order to evaluate accuracy results, average accuracies with k-fold cross-validation are calculated, for k=3, 5 and 10 folds.

Table 4. 3: K-fold accuracy results, for NB classifier with 3-gram features

| Folds | Label set | Accuracy |
|-------|-----------|----------|
| K = 3 | Groups | 0.76 |
| | Emotions | 0.63 |
| K = 5 | Groups | 0.78 |
| | Emotions | 0.63 |
| K=10 | Groups | 0.77 |
| | Emotions | 0.65 |

K-fold cross validation has verified the previous accuracy results. The accuracies have not changed much, for that reason it can be said that the position of instances in the dataset do not have a direct effect on classifiers.

Confusion matrixes for groups and emotions are presented below. Results are for the test dataset, and classifier trained with 3-grams are used.

Table 4. 4: Confusion matrix for emotions, NB classifier

| | Anger | Happiness | Love | Sadness |
|-----------|-------------|-------------|-------------|-------------|
| Anger | 1432 | 230 | 144 | 194 |
| Happiness | 252 | 1201 | 298 | 249 |
| Love | 128 | 248 | 1343 | 281 |
| Sadness | 188 | 321 | 215 | 1276 |

Table 4. 5: Confusion matrix for groups, NB classifier

| | Negative | Positive |
|----------|-------------|-------------|
| Negative | 3529 | 471 |
| Positive | 920 | 3080 |

According to the tables above, average precision, recall and f-score is calculated in table 4.6, in terms of micro and macro average.

Table 4. 6: Average precision-recall and f-scores, NB classifier

| | Method | Precision | Recall | F-score |
|----------|---------------|-----------|--------|---------|
| Emotions | micro-average | 0.618 | 0.619 | 0.618 |
| | macro-average | 0.651 | 0.621 | 0.633 |
| Groups | micro-average | 0.773 | 0.773 | 0.772 |
| | macro-average | 0.772 | 0.773 | 0.773 |

The micro and macro averaged values for precision, recall and f-score is very close to the accuracy results. The balanced number of instance sizes for each label causes very close outputs. Macro averaged result of precision for emotions seems a little higher, however, for multi-class problems micro average must be preferred as a metric, because it considers the imbalance between true positives of the classes.

4.1.2.2 Support Vector Machine

Two kinds of support vector machine classifiers are used in this study, the linear and non-linear SVM. Both classifiers are trained with n=1,2,3-gram features, and evaluated with training, test and manually labeled set.

Accuracy scores of linear and non-linear SVMs are calculated separately, in table 4.7.

Table 4. 7: Accuracy results for emotions, SVM classifiers

| Classifier | Features | Training Set | Test Set | Manually Labeled Set |
|----------------|----------|--------------|----------|----------------------|
| Non-Linear SVM | 1-gram | 0.85 | 0.55 | 0.55 |
| | 2-gram | 0.91 | 0.56 | 0.54 |
| | 3-gram | 0.94 | 0.58 | 0.57 |
| Linear SVM | 1-gram | 0.94 | 0.56 | 0.55 |
| | 2-gram | 0.95 | 0.60 | 0.59 |

| | | | |
|--------|------|------|------|
| 3-gram | 0.97 | 0.62 | 0.63 |
|--------|------|------|------|

According to the table above, linear SVM classifier (SVML) has performed better than non-linear one. But it is also exposed that in terms of accuracy, SVM classifiers have fallen behind the NB classifier.

For the same SVM classifiers, accuracy scores for groups are shown below, in table 4.8. The average accuracies with k-fold cross-validation are calculated, for k=3, 5 and 10 folds, in table 4.9.

In overall, maximum accuracy is 0.77 for groups and 0.63 for emotions. K-fold accuracy verifies the accuracy results by outputting close results.

Table 4. 8: Accuracy results for groups, SVM classifier

| Classifier | Features | Training Set | Test Set | Manually Labeled Set |
|-------------------|----------|--------------|----------|----------------------|
| Non-Linear SVM | 1-gram | 0.91 | 0.71 | 0.72 |
| | 2-gram | 0.95 | 0.74 | 0.72 |
| | 3-gram | 0.97 | 0.76 | 0.73 |
| Linear SVM (SVML) | 1-gram | 0.94 | 0.74 | 0.74 |
| | 2-gram | 0.95 | 0.75 | 0.76 |
| | 3-gram | 0.97 | 0.77 | 0.77 |

Table 4. 9: K-fold accuracy results, SVM classifiers with 3-gram features

| Folds | Label Set | Linear SVM | Non-Linear SVM |
|--------|-----------|------------|----------------|
| K = 3 | Groups | 0.73 | 0.75 |
| | Emotions | 0.57 | 0.56 |
| K = 5 | Groups | 0.74 | 0.75 |
| | Emotions | 0.61 | 0.58 |
| K = 10 | Groups | 0.77 | 0.77 |
| | Emotions | 0.63 | 0.60 |

Confusion matrixes for Linear-SVM with 3-gram features are presented below, in tables 4.10 and 4.11 since it has been found to be the most successful among trained SVM models. Results are collected with the test dataset.

Table 4. 10: Confusion matrix for emotions, SVM classifier

| | Anger | Happiness | Love | Sadness |
|-----------|-------------|------------|-------------|-------------|
| Anger | 1409 | 253 | 181 | 157 |
| Happiness | 257 | 933 | 494 | 316 |
| Love | 95 | 359 | 1565 | 281 |
| Sadness | 182 | 248 | 278 | 1292 |

Table 4. 11: Confusion matrix for groups, SVM classifier

| | Negative | Positive |
|----------|-------------|-------------|
| Negative | 2983 | 1017 |
| Positive | 805 | 3256 |

According to the tables above, average precision, recall and f-score is calculated in table 4.12, micro and macro averaged.

Table 4. 12: Average precision-recall and f-scores, SVM classifier

| | Method | Precision | Recall | F-score |
|----------|---------------|-----------|--------|---------|
| Emotions | micro-average | 0.600 | 0.600 | 0.600 |
| | macro-average | 0.595 | 0.620 | 0.615 |
| Groups | micro-average | 0.766 | 0.766 | 0.766 |
| | macro-average | 0.760 | 0.778 | 0.768 |

According to the table 4.12 macro-averaged results are higher than the micro averaged ones. Precision and recall are very close; displaying that number of false positives are still close to number of false negatives. However, recall is higher, meaning false negatives are slightly less in numbers.

So far, Naïve Bayes classifiers have output better results in all metrics; accuracy, precision, recall and f-score.

4.1.2.3 FastText Classifier

FastText classifiers are trained with $n=1, 2, 3$ grams, as the previous classifiers have been. Two supervised classifiers are utilized, one is trained with the pretrained Turkish word embeddings provided by FastText, the other one is trained with vectors consisted of word embeddings of the tweet corpus's training data.

Table 4.13 provides accuracy scores for emotions, and table 4.14 provides accuracy scores for groups.

Table 4. 13: Accuracy results for emotions, FastText classifier

| Vector Set | Features | Training Set | Test Set | Manually Labeled Set |
|--------------------|----------|--------------|----------|----------------------|
| Pretrained Turkish | 1-gram | 0.98 | 0.66 | 0.72 |
| | 2-gram | 0.98 | 0.68 | 0.72 |
| | 3-gram | 0.99 | 0.70 | 0.73 |
| Tweet Corpus | 1-gram | 0.87 | 0.63 | 0.68 |
| | 2-gram | 0.97 | 0.67 | 0.69 |
| | 3-gram | 0.97 | 0.68 | 0.70 |

Table 4. 14: Accuracy results for groups, FastText classifier

| Vector Set | Features | Training Set | Test Set | Manually Labeled Set |
|--------------------|----------|--------------|----------|----------------------|
| Pretrained Turkish | 1-gram | 0.97 | 0.78 | 0.81 |
| | 2-gram | 0.97 | 0.80 | 0.82 |
| | 3-gram | 0.99 | 0.81 | 0.84 |
| Tweet Corpus | 1-gram | 0.97 | 0.75 | 0.74 |
| | 2-gram | 0.99 | 0.76 | 0.76 |
| | 3-gram | 0.99 | 0.77 | 0.77 |

According to the results above, FastText classifier has given the best results so far. Accuracy of emotions have reached 70 percents; 70 percent in test data and 73 percent in the manually labeled data. For groups, the results are even higher, reaching to 84 percent with pretrained Turkish embeddings.

Confusion matrix for the FastText classifier trained with 3-grams embeddings and pretrained vectors is as the following, in table 4.15 and 4.16.

Table 4. 15: Confusion matrix for emotions, FastText classifier

| | Anger | Happiness | Love | Sadness |
|-----------|-------------|-------------|-------------|-------------|
| Anger | 1437 | 269 | 95 | 190 |
| Happiness | 234 | 1151 | 372 | 250 |
| Love | 95 | 400 | 1252 | 199 |
| Sadness | 218 | 273 | 225 | 1340 |

Table 4. 16: Confusion matrix for groups, FastText classifier

| | Negative | Positive |
|----------|-------------|-------------|
| Negative | 3231 | 816 |
| Positive | 828 | 3125 |

The number of true positives for emotions do not differ very much compared to the previous classifiers, however for both *negative* and *positive* groups, number of true positives are higher. Therefore, it is expected precision and recall being higher.

Table 4. 17: Average precision-recall and f-scores, FastText classifier

| | Method | Precision | Recall | F-score |
|----------|---------------|-----------|--------|---------|
| Emotions | micro-average | 0.647 | 0.647 | 0.647 |
| | macro-average | 0.648 | 0.653 | 0.650 |
| Groups | micro-average | 0.794 | 0.794 | 0.794 |
| | macro-average | 0.792 | 0.804 | 0.797 |

As it can be seen in table 4.17, precision and recall are higher than the previous results, by 0.03 - 0.04. Although this is a minor difference, these are the best observed results in the study.

4.1.2.4 Convolutional Neural Network

A single convolutional neural network has been trained with tweet corpus data, and word embeddings are provided by FastText's pretrained Turkish dataset. Accuracy scores for emotions and groups are presented in table 4.17 and 4.18, respectively.

Table 4. 18: Accuracy results for emotions, CNN classifier

| Classifier | Features | Training Set | Test Set | Manually Labeled Set |
|------------|----------|--------------|----------|----------------------|
| CNN | 1-gram | 0.82 | 0.60 | 0.61 |
| | 2-gram | 0.82 | 0.61 | 0.65 |
| | 3-gram | 0.84 | 0.63 | 0.67 |

Table 4. 19: Accuracy results for groups, CNN classifier

| Classifier | Features | Training Set | Test Set | Manually Labeled Set |
|------------|----------|--------------|----------|----------------------|
| CNN | 1-gram | 0.92 | 0.72 | 0.71 |
| | 2-gram | 0.93 | 0.74 | 0.76 |
| | 3-gram | 0.95 | 0.77 | 0.78 |

CNN classifier has output close accuracy results to NB and SVM, but still drops behind FastText classifier. 3-grams are still the most successful features.

Confusion matrixes for the most successful CNN model, which is the one trained with 3-gram features, is presented below, in table 4.19 and 4.20.

Table 4. 20: Confusion matrix for emotions, CNN classifier

| | Anger | Happiness | Love | Sadness |
|-----------|-------------|-------------|-------------|-------------|
| Anger | 1412 | 209 | 233 | 146 |
| Happiness | 190 | 1403 | 301 | 106 |
| Love | 80 | 331 | 1311 | 278 |
| Sadness | 393 | 226 | 147 | 1234 |

Table 4. 21: Confusion matrix for groups, CNN classifier

| | Negative | Positive |
|----------|-------------|-------------|
| Negative | 3258 | 742 |
| Positive | 1058 | 2942 |

According to the confusion matrixes above, precision and recall values are calculated and presented in table 4.21.

Table 4. 22: Average precision-recall and f-scores, CNN classifier

| | Method | Precision | Recall | F-score |
|----------|---------------|-----------|--------|---------|
| Emotions | micro-average | 0.656 | 0.656 | 0.656 |
| | macro-average | 0.676 | 0.665 | 0.671 |
| Groups | micro-average | 0.772 | 0.772 | 0.772 |
| | macro-average | 0.778 | 0.771 | 0.769 |

To sum up, CNN has output close results to SVM and NB, but has higher precision, recall and f-score. This is due to higher results of true positives, but also lower false positives and false negatives. Therefore, CNN can be considered as the best classifier after FastText, in this study.

4.1.3 Summary

According to the results, overall accuracy is approximately 65% for emotions and 78% for groups. Accuracy results can be considered reliable; because the data is distributed evenly across all classes and k-fold cross validation also verifies the scores.

Since the proposed approach in this study is to use emojis as features for sentiment analysis, manually labeled data are evaluated alongside the test data. Consequently, in all classifiers, the evaluation results of the manually tagged data were very close to the test data, therefore proving the objectives of the study.

In terms of accuracy, the most successful classifier has been the FastText classifier. Naive Bayes and SVM managed to output similar results to each other and have given the best possible results, through the utilization of n-gram features. Convolutional Neural Network has seemed to lag behind FastText, however it should be noted that CNN offers more customization and tuning possibilities than other classifiers, and it is possible that CNN would provide better results for this problem, with a different configuration.

The evenly distribution of the data across all classes, this has also resulted in close precision-recall results. The proximity between precision and recall means false negatives and false positives are very close in numbers. As a complete metric, F-score has been calculated and it has been highest in the FastText classifier, as in the accuracy.

Confusion matrixes show that there are overlaps between classes. Overlaps between emotions on same groups are reasonable due to the semantic margin, e.g. love and happiness; and this is the case of as it can be seen in confusion matrixes.

Per-class precisions and recalls are not calculated; however, classifiers' performances across emotion classes can be traced on confusion matrixes. In overall, the classifiers are the most successful when classifying *anger*; probably because angry sentences contain more unique and characteristic terms, like curse words. On the other hand, *happiness* has

the lowest number of true positives; and this is probably due to social media posts containing happy emojis are more frequent and contain more common words.



5. CONCLUSION

Recent studies in sentiment analysis reveal that detecting or directing feelings or opinions become a cutting edge in social complex networks, information security systems, multimodal semantics and media mining. The challenge of sentiment classification is related to the expressiveness of languages. In current technologies, emojis become a standard way to express feelings for language dependent context. Moreover, they help gaining insight on people's opinions on different topics. Turkish language bears its own challenges because of its structure.

About preceding studies similar to ours, we remark that sentiment analysis was performed using domain dependent datasets. Very few studies have developed a generic solution for domain independent datasets and the size of datasets in these studies were very small. In this study, we wanted to develop a generic emotion classifier for Turkish texts, while using the emoji to direct the emotion detecting and data labeling processes.

In order to do such a study, we gathered a comparatively large dataset; we applied several preparation steps and extracted features from this dataset. Then we trained and evaluated classifiers that had been proven to be efficient in text classification studies. The evaluation results seemed a bit lower than the results in preceding studies; emojis show high accuracy rates in the sentiment classification through positive or negative sentiment features, when there are two classes for sentiments. On the other hand, classifying the emotions without grouping them as positive or negative, instead using the emotion itself for classification did not present satisfactory results. When there were four classes, performance of classifiers actually lowered.

In spite of lower results compared to previous studies, we have been successful in implementing a generic classification. Evaluation results with manually labeled data have output very close results to the data labeled by emojis. This actually means that using

emojis to identify sentiments in texts is actually accurate and very close to the labeling done by humans. The main assertion of the study was proven to be accurate.

It must be noted that, even though the data we had seemed sufficient in number and quality at first glance, we had to discard some of it in later steps, because of noise and inaccuracy. This is a typical characteristics of social media data, which must always be considered in any study dependent on social media data. In order to do more accurate studies for similar purposes, larger and more coherent datasets should be used, which may be labeled by emojis or manually. The dataset obtained and used in this study will also be available to the researchers and thus be used as a core for future studies.

The feature extraction models utilized in this study; bag-of-words and word embeddings, have proved themselves successful in most of the NLP studies so far, but they may still miss some modalities in written language. In the results, we could note that the most informative features are spot-on; like *korkak* (*coward*) informative to the class anger or *ağla* (*cry*) informative to the class sadness. On the other hand, to perform a more accurate analysis, we have to address the difference between *Mutsuzum - I am unhappy* and *Mutsuz olduğum söylenemez - It cannot be said that I am unhappy*. This form becomes hard to resolve when word or vector features have been used, even if they are provided with their n-grams. Thus, it is needed to be found methods to extract features from inverted sentences or indirect expressions, respecting the meaning of the sentence.

5.1 Limitations and Future Work

Neural network architectures other than CNN also proved themselves successful in text classification. Recurrent Neural Networks (RNN), which is a type of neural network architecture, are trained in loops, allowing persisting previous input and adjusting the next step depending on the result of previous step. A special implementation of RNN is Long-Short Term Memory (LSTM), which solves the long-term dependency problem by remembering important inputs from distant previous steps. In Turkish text classification studies, there has not been any implementation of RNN, therefore it needs to be tried and explored in further studies.

In order to adapt Turkish language to NLP studies better, it is necessary to have toolsets that perform semantic and morphological analysis, supported by lexicons or treebanks. Extracting better features from Turkish will gain importance as machine learning and natural language processing studies penetrate into everyday life.

The expression of emotions by people is closely related to the words that emphasize subjectivity within the sentence. The order of the words in the sentence is equally important. In order to extract these features from the sentence and make them be used in the sentiment analysis, feature extraction methods other than the lexical parsing methods should be used.



REFERENCES

- Akın, A.A. & Akın, M.D., *Zemberek, an open source nlp framework for Turkic languages.* *Structure*, 2007. Available at: http://zemberek.googlecode.com/files/zemberek_makale.pdf
- Alpaydın, E., *Yapay Öğrenme*, Boğaziçi Üniv. Yayınları, 2011
- Amasyalı, M. F., & Diri, B. (2006, May). Automatic Turkish text categorization in terms of author, genre and gender. In *International Conference on Application of Natural Language to Information Systems* (pp. 221-226). Springer, Berlin, Heidelberg.
- Ayvaz, Serkan & O. Shiha, Mohammed. (2017). The Effects of Emoji in Sentiment Analysis. *International Journal of Computer and Electrical Engineering*. 9. 360-369. 10.17706/IJCEE.2017.9.1.360-369.
- Boynukalin, Z. (2012). Emotion analysis of Turkish texts by using machine learning methods. *Middle East Technical University*.
- Dehkharghani, R., Saygin, Y., Yanikoglu, B., & Oflazer, K. (2016). SentiTurkNet: a Turkish polarity lexicon for sentiment analysis. *Language Resources and Evaluation*, 50(3), 667-685.
- Eisner, B., Rocktäschel, T., Augenstein, I., Bošnjak, M., & Riedel, S. (2016). emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.
- Eryiğit, G., & Adalı, E. (2004, February). An affix stripping morphological analyzer for Turkish. In *Proceedings of the IASTED international conference artificial intelligence and applications*.
- Gezici, Gizem & Yanikoglu, Berrin. (2018). Sentiment Analysis in Turkish. 10.1007/978-3-319-90165-7_12.
- Goodfellow, I., Bengio, Y., Courville, A., *Deep Learning*, MIT Press, 2016
- Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

- Jurafsky, D., Martin J.H., *Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall; 2nd edition (May 16, 2008)
- Kaya, M., Fidan, G., & Toroslu, I. H. (2012, December). Sentiment analysis of turkish political news. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01* (pp. 174-180). IEEE Computer Society.
- Li, X., Yan, R., & Zhang, M. (2017, July). Joint emoji classification and embedding learning. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data* (pp. 48-63). Springer, Cham.
- Meral, M., & Diri, B. (2014, April). Sentiment analysis on Twitter. In *2014 22nd Signal Processing and Communications Applications Conference (SIU)* (pp. 690-693). IEEE.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).
- H. Mulki, H. Haddad, C. B. Ali and İ. Babaoğlu, "Preprocessing impact on Turkish sentiment analysis," *2018 26th Signal Processing and Communications Applications Conference (SIU)*, Izmir, 2018, pp. 1-4.
- Novak, P. K., Smailović, J., Sluban, B., & Mozetič, I. (2015). Sentiment of emojis. *PloS one*, *10*(12), e0144296.
- Oflazer, Kemal & Saraclar, Murat. (2018). Turkish and Its Challenges for Language and Speech Processing. 10.1007/978-3-319-90165-7_1.
- Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).
- Sak, H., Güngör, T., & Saraçlar, M. (2011). Resources for Turkish morphological processing. *Language resources and evaluation*, *45*(2), 249-261.
- Türkmenoglu, C., & Tantug, A. C. (2014, June). Sentiment analysis in Turkish media. In *International Conference on Machine Learning (ICML)*.

- Vural, A. G., Cambazoglu, B. B., Senkul, P., & Tokgoz, Z. O. (2013). A framework for sentiment analysis in turkish: Application to polarity detection of movie reviews in turkish. In *Computer and Information Sciences III* (pp. 437-445). Springer, London.
- Yıldırım, E., Çetin, F. S., Eryiğit, G., & Temel, T. (2015). The impact of NLP on Turkish sentiment analysis. *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi*, 7(1), 43-51.



APPENDICES

Appendix A

A.1 Filtered Word List

| | | |
|----------------|---------------|----------------|
| @ | federasyon | namjoon |
| agüero | fener | reklam yapılır |
| akparti | fenerbahçe | rt @ |
| anestezimağdur | galatasaray | seçim |
| bahçeli | galibiyet | slimani |
| belhanda | gülriz | suriyeliler |
| beşiktaş | http | sururi |
| bingölspor | imamoğlu | tamam |
| büyükşehir | jeon | tayyip |
| comolli | jimin | uygunfiyata |
| cvp | joon | volkan |
| devam | kılıçdaroğlu | çakar |
| ensobette | maç | şampiyonluk |
| erdoğan | millisavunma | |
| ersun | mujdebekliyor | |

A.2 Turkish Stopword List

| | | | | |
|--------|---------|---------|----------|-------|
| a | ancak | aslında | bazı | beni |
| acaba | arada | ayrıca | bazıları | benim |
| altı | artık | az | belki | beri |
| altmış | asla | bana | ben | beş |
| ama | aslında | bazen | benden | bile |

| | | | | |
|-----------|-------------|-------------|-------------|------------|
| bilhassa | dolayı | hiçbir | nereye | öyle |
| bin | dolayısıyla | hiçbiri | neyse | oysa |
| bir | dört | i | niçin | pek |
| biraz | e | ı | nin | rağmen |
| birçoğu | edecek | için | nın | sana |
| birçok | eden | içinde | niye | sanki |
| biri | ederek | iki | nun | sanki |
| birisi | edilecek | ile | nün | şayet |
| birkaç | ediliyor | ilgili | o | şekilde |
| birşey | edilmesi | ise | öbür | sekiz |
| biz | ediyor | işte | olan | seksen |
| bizden | eğer | itibaren | olarak | sen |
| bize | elbette | itibariyle | oldu | senden |
| bizi | elli | kaç | olduğu | seni |
| bizim | en | kadar | olduğunu | senin |
| böyle | etmesi | karşın | olduklarını | şey |
| böylece | etti | kendi | olmadı | şeyden |
| bu | ettiği | kendilerine | olmadığı | şeye |
| buna | ettiğini | kendine | olmak | şeyi |
| bunda | fakat | kendini | olması | şeyler |
| bundan | falan | kendisi | olmayan | şimdi |
| bunlar | filan | kendisine | olmaz | siz |
| bunları | gene | kendisini | olsa | siz |
| bunların | gereği | kez | olsun | sizden |
| bunu | gerek | ki | olup | sizden |
| bunun | gibi | kim | olur | size |
| burada | göre | kime | olur | sizi |
| bütün | hala | kimi | olursa | sizi |
| çoğu | halde | kimin | oluyor | sizin |
| çoğunu | halen | kimisi | on | sizin |
| çok | hangi | kimse | ön | sonra |
| çünkü | hangisi | kırk | ona | şöyle |
| da | hani | madem | önce | şu |
| daha | hatta | mi | ondan | şuna |
| dahi | hem | mı | onlar | şunları |
| dan | henüz | milyar | onlara | şunu |
| de | hep | milyon | onlardan | ta |
| defa | hepsi | mu | onları | tabii |
| değil | her | mü | onların | tam |
| diğer | herhangi | nasıl | onu | tamam |
| diğeri | herkes | ne | onun | tamamen |
| diğerleri | herkese | neden | orada | tarafından |
| diye | herkesi | nedenle | öte | trilyon |
| doksan | herkesin | nerde | ötürü | tüm |
| dokuz | hiç | nerede | otuz | tümü |

| | | | | |
|-------|---------|------------|--------|-------|
| u | vardı | yapılması | ye | yirmi |
| ü | ve | yapıyor | yedi | yoksa |
| üç | veya | yapmak | yerine | yu |
| un | ya | yaptı | yetmiş | yüz |
| ün | yani | yaptığı | yi | zaten |
| üzere | yapacak | yaptığını | yı | zira |
| var | yapılan | yaptıkları | yine | |

A.3 Synonym List

| | | |
|--------------------|------------------|-----------------------|
| ad, isim | cihaz, aygıt | fakir, yoksul, fukara |
| adale, kas | cömert, eliaçık | fert, birey |
| ahenk, uyum | cümle, tümce | gelecek, istikbal |
| alaka, ilgi | çabuk, acele | genel, umumi |
| anlam, mana | çağdaş, modern | güz, sonbahar |
| atamak, tayin | çehre, yüz | hadise, olay |
| ayraç, parantez | dahil, iç | irmak, nehir |
| ayrıcılık, imtiyaz | dargın, küs | ihtiyar, yaşlı |
| barış, sulh | delil, kanıt | ilim, bilim |
| baş, kafa | deney, tecrübe | itibar, saygınlık |
| batı, garp | dergi, mecmua | izah, açıklama |
| baytar, veteriner | dil, lisan | kalite, nitelik |
| beden, gövde | doğa, tabiat | kanun, yaza |
| belge, vesika | doğu, şark | karşıt, zıt |
| bellek, hafıza | düş, rüya | keder, üzüntü |
| besin, gıda | ebedi, sonsuz | kelime, sözcük |
| biçim, şekil | ehemmiyet, önem | kişi, şahıs |
| bilgin, alim | ek, ilave | kolay, basit |
| birdenbire, aniden | ekonomi, iktisat | konuk, misafir |
| buyruk, emir | esas, temel | lakin, ama, fakat |
| cenk, savaş | esir, tutsak | lüzumlu, gerekli |
| cevap, yanıt | etraf, çevre | lüzumsuz, gereksiz |

mahpushane, cezaevi
mebus, milletvekili
mecbur, zorunlu
medeniyet, uygarlık
meridyen, boylam
mesela, örneğin
mesele, sorun, problem
millet, ulus
mübarek, kutsal
müessese, kuruluş
müspet, olumlu
neden, sebep
nesil, kuşak
noksan, eksik
okul, mektep
olanak, imkan
olası, mümkün
olay, vaka
olumsuz, menfi
ömür, hayat, yaşam
öneri, teklif
önlem, tedbir
örgüt, teşkilat
özgün, orijinal
özgür, hür
özlem, hasret
pabuç, ayakkabı
politika, siyaset
rastlantı, tesadüf
rüzgar, yel

samimi, içten
sanayi, endüstri
sınav, imtihan
suni, yapay
sürat, hız
şaka, latife
şans, talih
şart, koşul
şayet, eğer
şef, lider, önder
tabip, hekim, doktor
tercüme, çeviri
tertip, düzen
uçak, tayyare
ufak, küçük
ulu, yüce
ümit, umut
ünite, birim
ünlü, meşhur
vakit, zaman
varlıklı, zengin
vasıta, araç
yaratık, mahluk
yardımcı, muavin
yargıç, hakim
yerel, mahalli
yetenek, kabiliyet
yıl, sene
yine, tekrar
yüzyıl, asır

zarar, ziyan

BIOGRAPHICAL SKETCH

Çağatay Ünal YURTÖZ was born in 1988, Zonguldak. He graduated from Ege University Department of Computer Engineering in 2012 and he has been working as a software developer in private companies for 7 years. He began master's studies in Galatasaray University in 2016 and has been studying on machine learning and natural language processing since.

PUBLICATIONS

Ç. Ü. Yurtoz and I. B. Parlak, "Measuring the effects of emojis on Turkish context in sentiment analysis," *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, Barcelos, Portugal, 2019, pp. 1-6.