**Twitter Sentiment Analysis**

Filiz Dalıp, Akif Çetin, Ömer Yasin Kolkanat, Murat Can Ganiz

Department of Computer Engineering

Marmara University

İstanbul, Turkey

{filizdalip, akf.ctn, omerkolkanat, mcganiz}@gmail.com

*Abstract*— Sentiment Analysis is the process of determining whether a piece of writing is positive and negative. It’s also known as opinion mining. In this project, We collected tweets using Twitter API and we labelled 1076 unlabeled tweets manually. Then, we prepared with getting feature vector using words to machine learning algorithm for classification. We tried 8 different machine learning algorithms and we choosed Logistic Regression algorithm from these algorithms that accuracy is %79.

*(Abstract)*

Keywords-sentiment analysis; text mining;

# Introduction

Sentiment Analysis is the process of determining whether a piece of writing is positive and negative. It’s also known as opinion mining.

Twitter is a popular social network where users can share short SMS-like messages called tweets. Users share thoughts, links and pictures on Twitter, journalists comment on live events, companies promote products and engage with customers. The list of different ways to use Twitter could be really long, and with 500 millions of tweets per day, there’s a lot of data to analyze and to play with.

The purpose of this project is to build an algorithm that can accurately classify Turkish Tweets as positive or negative. Our hypothesis is that we can obtain high accuracy on classifying sentiment in tweets using machine learning algorithms.

We labeled all of tweets that from gained with Twitter API and there are some tweet examples like below.

|  |  |
| --- | --- |
| **Tweet** | **Sentiment** |
| Bugün hava çok güzel. | Positive |
| Sınavım kötü geçti. | Negative |
| Hayat çok zor. | Negative |

Figure 1. Tweet examples with sentiments.

We get all tweets in json format. There are a lot of information in the format like below. But, we focused just text.

The key attributes are the following:

text: the text of the tweet itself

created\_at: the date of creation

favorite count, retweet count: the number of favorites and retweets

favorited, retweeted: boolean stating whether the authenticated user (you) have favorited or retweeted this tweet

lang: acronym for the language (e.g. “en” for English)

id: the tweet identifier

place, coordinates, geo: geo-location information if available

user: the author’s full profile

entities: list of entities like URLs, @-mentions, hashtags and symbols

in\_reply\_to\_user\_id: user identifier if the tweet is a reply to a specific user

in\_reply\_to\_status\_id: status identifier id the tweet is a reply to a specific status

We start our analysis by breaking the text down into words. Tokenization is one of the most basic, yet most important, steps in text analysis. The purpose of tokenization is to split a stream of text into smaller units called tokens, usually words or phrases. While this is a well understood problem with several out-of-the-box solutions from popular libraries, Twitter data pose some challenges because of the nature of the language.

We didn’t use NLTK because NLTK doesn’t support Turkish words. Therefore, we did data preprocessing manually using regular expression library in Python. Also we defined stop words in Turkish language and we extract them from all of tweets.

The rest of the paper is organized as follows. The related work and background are covered in section II. We then present our approach for focused Twitter sentiment analysis optimization in section III. We follow this by summarizing our experimental results in section IV. Section V includes a conclusion and discusses the future work.

# Related Work and background

Sentiment analysis is a growing area of Natural Language Processing with research ranging from document level classification (Pang and Lee 2008) to learning the polarity of words and phrases (e.g., (Hatzivassiloglou and McKeown 1997; Esuli and Sebastiani 2006)). Given the character limitations on tweets, classifying the sentiment of Twitter messages is most similar to sentence-level sentiment analysis (e.g., (Yu and Hatzivassiloglou 2003; Kim and Hovy 2004)); however, the informal and specialized language used in tweets, as well as the very nature of the microblogging domain make Twitter sentiment analysis a very different task. It’s an open question how well the features and techniques used on more well-formed data will transfer to the microblogging domain.

Just in the past year there have been a number of papers looking at Twitter sentiment and buzz (Jansen et al. 2009; Pak and Paroubek 2010; O’Connor et al. 2010; Tumasjan et al. 2010; Bifet and Frank 2010; Barbosa and Feng 2010; Davidov, Tsur, and Rappoport 2010). Other researchers have begun to explore the use of part-of-speech features but results remain mixed. Features common to microblogging (e.g., emoticons) are also common, but there has been little investigation into the usefulness of existing sentiment resources developed on non-microblogging data.

Researchers have also begun to investigate various ways of automatically collecting training data. Several researchers rely on emoticons for defining their training data (Pak and Paroubek 2010; Bifet and Frank 2010). (Barbosa and Feng 2010) exploit existing Twitter sentiment sites for collecting training data. (Davidov, Tsur, and Rappoport 2010) also use hashtags for creating training data, but they limit their experiments to sentiment/non-sentiment classification, rather than 3-way polarity classification, as we do.

# Approach

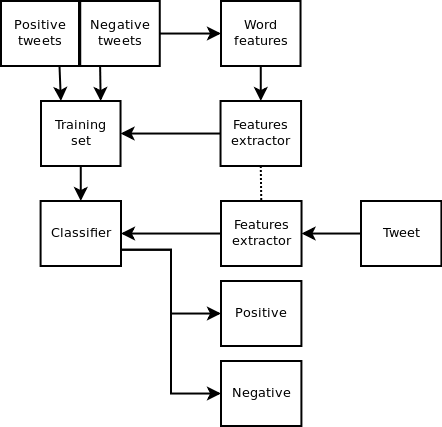


Figure 2. Sentiment Analysis Flow Chart

We labeled 1076 tweets as positive and negative.

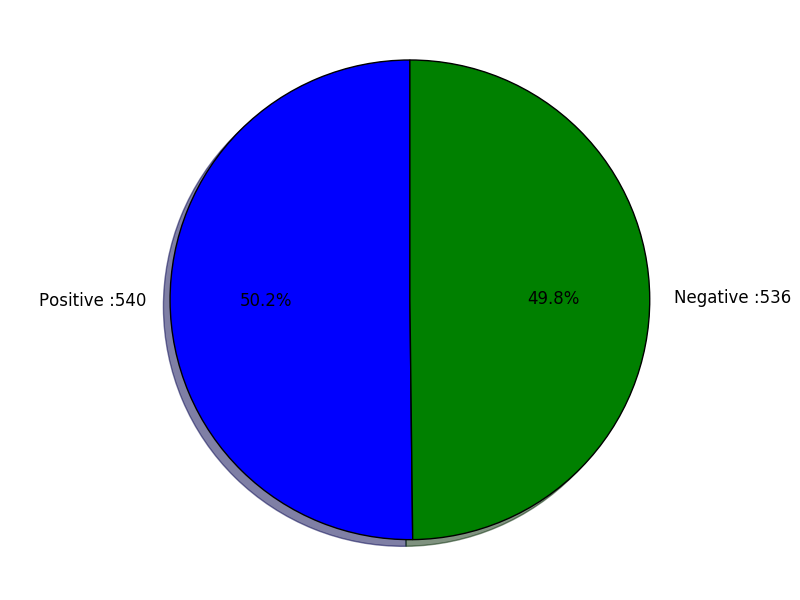


Figure 3. Our dataset category percentages

Before data pre-processing part, there are top 10 words for negative and positive dirty data.

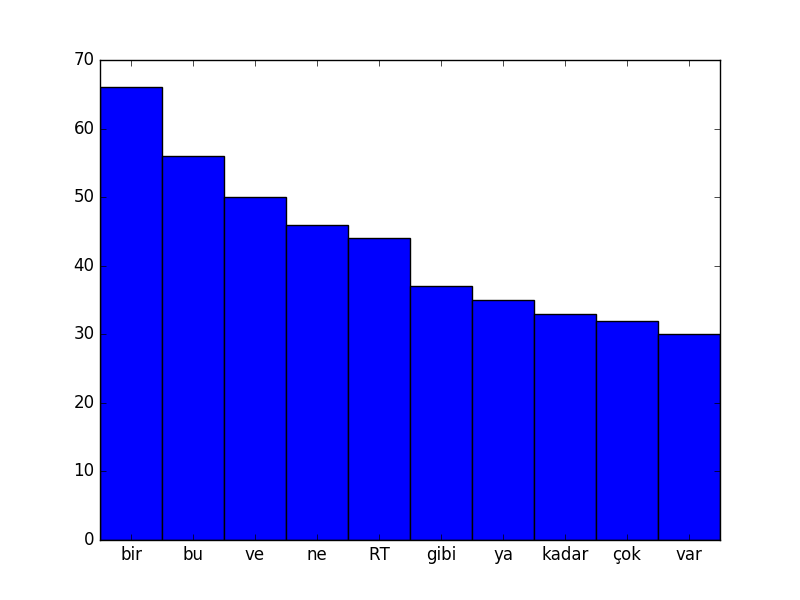


Figure 4. Top 10 words for Negative Dirty Data

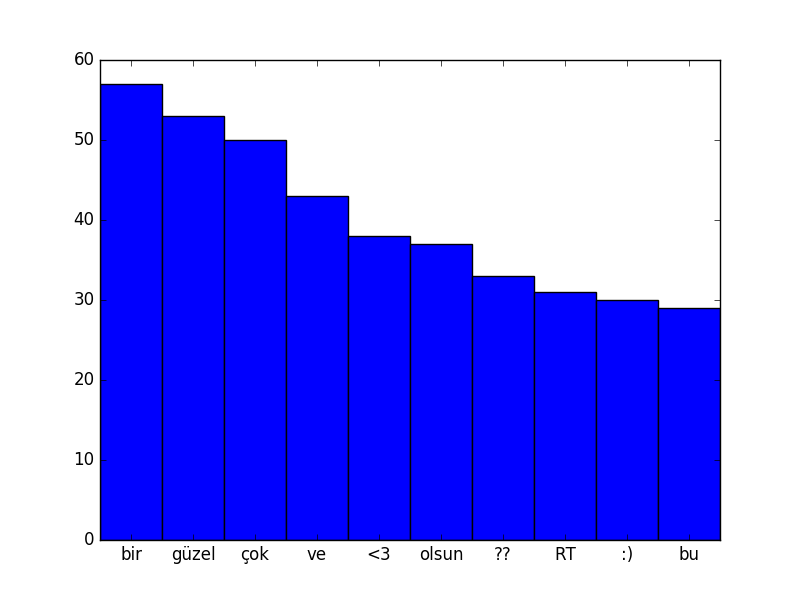


Figure 5. Top 10 words for Positive Dirty Data

We should clean this dirty data because, there are meaningless words and common words for both positive and negatives so we cannot use them. For this situation, we used regular expression library in Python to clean our dirty data. Also, we rejected Turkish stop words from our dataset.

After data pre-processing part dataset is ready for machine learning algorithms and top 10 words for our clean data is like below.

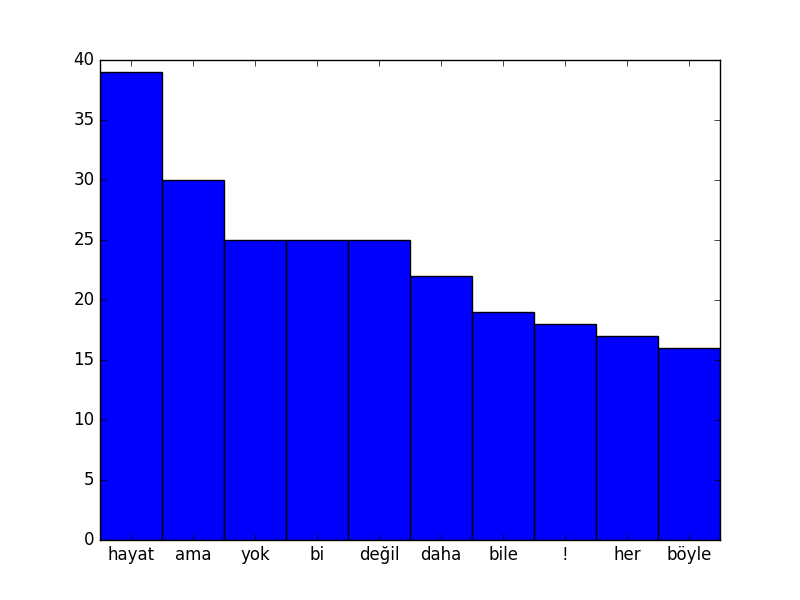


Figure 6. Top 10 words for Negative Clean Data

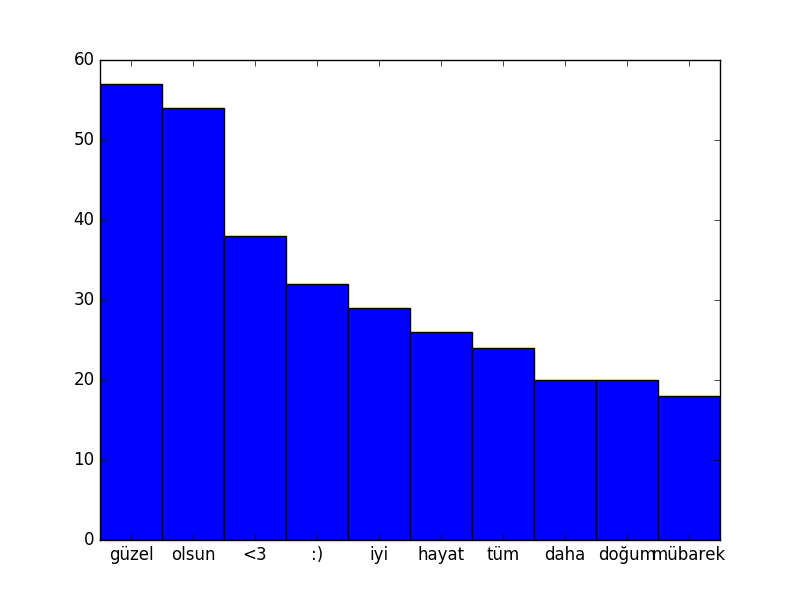


Figure 7. Top 10 words for Positive Clean Data

# Expırement Setup

We used scikit-learn library in Python for implementation of machine learning algorithm. We tried 8 different machine learning algorithms like below.

1) Logistic Regression

2) Naïve Bayes

3) Multinomial Naïve Bayes

4) Bernoulli Naïve Bayes

5) Linear Support Vector Classification

6) Stochastic Gradient Descent Classifier

7) Decision Tree

8) Random Forest

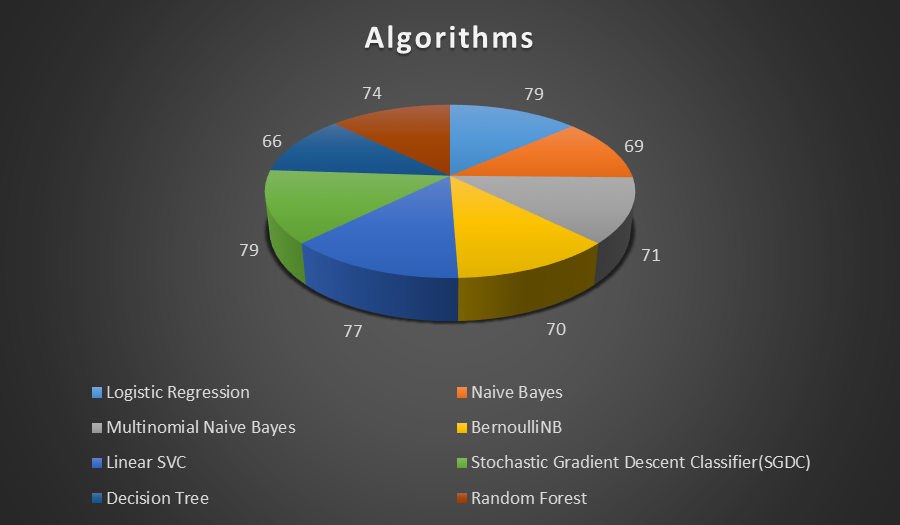
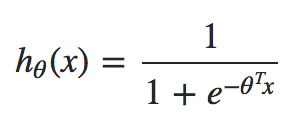


Figure 8. Algorithm accuracy pie chart

We choosed Logistic Regression algorithm for giving better result generally.

We train n logistic classifiers on our training data, with each classifier producing as output the probability that a given sample is a member of the class the classifier is trained to recognize. Once training is complete, to classify a new sample we let all the trained classifiers try to recognize it and accept the judgement of the classifier that produces the highest probability value. This is known as one-vs-all or one-vs-rest classification and is a standard way to use binary classifiers to do multiclass classification.

The weights are applied the same way in classifying new data as they are when we are training the classifier. We have a weight vector θ containing coefficients that when combined with a sample's feature vector x yield an hypothesis value. For linear regression θTx (expanded: θ0+θ1x1+θ2x2+...) yields the hypothesis value. For logistic regression θTx is used as the exponent in the logistic function to produce the hypothesis value



e is the usual natural log constant ≈ 2.71828.

hθ(x) is the probability that the sample x is a member of class the classifier recognizes. Compute hθ(x) for each classifier and accept the verdict of the classifier that produces the largest probability value.

Gradient descent works the same way for logistic regression as it does for linear regression. We are still trying to minimize the cost function by iteratively nudging the weights to better values using partial derivatives of the cost function. The hypothesis function is different for logistic regression, but the way it is used in gradient descent is the same. Once we have written code to do gradient descent for linear regression, we should be able to just plug in a different hypothesis function and have it work for logistic regression.

Pseudocode for one iteration of gradient descent:

newtheta := theta;

learning\_rate := 0.01;

for k := 1 to n

sum := 0

for i := 1 to m

sum := sum + (hypothesis(x[i], theta) - y[i]) \* x[i][k];

end

nudge := sum \* learning\_rate;

newtheta[k] := newtheta[k] - nudge;

end

theta := newtheta;

**x** is a matrix containing our training data, one sample per row.

**y** is a vector containing the correct classification prediction for each sample, 1 if the sample is in the class, 0 otherwise.

**m** is the number of samples.

**n** is the number of features.

**Experiment Results :**

#### test-on-training

|  |  |
| --- | --- |
| Algorithm | Accuracy |
| Logistic Regression | 99.07 |
| Naïve Bayes | 98.14 |
| Multinomial Naïve Bayes | 99.07 |
| Bernoulli Naïve Bayes | 96.29 |
| Linear Support Vector Classification | 100 |
| Stochastic Gradient Descent Classifier | 100 |
| Decision Tree | 100 |
| Random Forest | 98.14 |

#### %10 testing, %90 training

|  |  |
| --- | --- |
| Algorithm | Accuracy |
| Logistic Regression | 79.62 |
| Naive Bayes | 69.44 |
| Multinomial Naive Bayes | 71.29 |
| Bernoulli Naive Bayes | 70.37 |
| Linear SVC | 77.77 |
| SGDC | 79.62 |
| Decision Tree | 66.66 |
| Random Forest | 74.07 |

#### 10-fold cross validation

|  |  |
| --- | --- |
| Algorithm | Accuracy |
| Logistic Regression | 78.16 |
| Naive Bayes | 76.25 |
| Multinomial Naive Bayes | 81.44 |
| Bernoulli Naive Bayes | 70.47 |
| Linear SVC | 77.22 |
| SGDC | 76.89 |
| Decision Tree | 71.38 |
| Random Forest | 72.83 |

# Conclusıon & Future Work

Sentiment analysis is an evolving field with a variety of

use applications. Although sentiment analysis tasks are

challenging due to their natural language processing

origins, much progress has been made over the last

few years due to the high demand for it. Not only do

companies want to know how their products and

services are perceived by consumers (and compare to

competitors), but consumers want to know the opinions

of others before making buying decisions.

We conclude that using different machine learning algorithms it is easier to classify the tweets and more we improve the training dataset to increase accuracy.

##### We look forward to use bigger dataset to increase the accuracy.

##### References

B. Jansen, M. Zhang, K. Sobel, A. Chowdury. The Commerical Impact of Social Mediating Technologies: Micro-blogging as Online Word-of-Mouth Branding, 2009.

O’Connor, B.; Balasubramanyan, R.; Routledge, B.; and

Smith, N. 2010. From tweets to polls: Linking text sentiment

to public opinion time series. In Proceedings of ICWSM.

C. Manning and H. Schuetze. Foundations of Statistical Natural Language Processing. 1999.

Pak, A., and Paroubek, P. 2010. Twitter as a corpus for

sentiment analysis and opinion mining. In Proc. of LREC.

Davidov, D.; Tsur, O.; and Rappoport, A. 2010. Enhanced

sentiment learning using twitter hashtags and smileys. In

Proceedings of Coling.

B. Pang, L. Lee, S. Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques, 2002.

B. Pang and L. Lee. "Opinion Mining and Sentiment Analysis" in Foundations and Trends in Information Retrieval, 2008.

B. Pang and L. Lee. "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts" in Proceedings of ACL, 2004.

Kim, S.-M., and Hovy, E. 2004. Determining the sentiment

of opinions. In Proceedings of Coling.

J. Read. Using Emotions to Reduce Dependency in Machine Learning Techniques for Sentiment Classification, 2005.

P. Turney. "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews" in Proceedings of the 40th Annual Meeting of the Association for Computatoinal Linguistics (ACL), 2002.