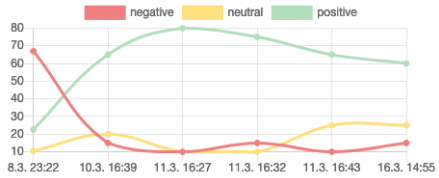
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2020
http://excel.fit.vutbr.cz

Estimation of Emotions from a Text and Analyzing Tweets

Aneta Dufková*



WHAT DOES TWITTER THINK .com

Abstract

This paper describes the process of estimation of emotions from a text using machine learning. Negative, positive, and neutral emotions are recognized from tweets focused on various topics. The whole work is not only about machine learning, but also about natural language processing. It involves data gathering, preprocessing of obtained texts, and a lot of experimenting with both model and dataset.

The final model has been used to create a simple web application whatdoestwitterthink.com, which allows user to discover what do people on Twitter think. User can write a topic, the app downloads tweets related to this topic in real time, and analyzes them. The application also collects feedback from users to improve the classifier.

Keywords: sentiment analysis — tweet sentiment — estimation of emotions

Supplementary Material: Demonstration Video — Website

*xdufko02@fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Estimation of emotions, in other words sentiment analysis, is a very important problem. There are many companies which would like to know what does the public think about them or their new product [1]. Hoteliers want to know whether their customers are satisfied with their services or not [2]. It is not necessary, however, to only look for examples in business. Everyone can sometimes be curious about what do other people think about some topic, and this work aims to provide a simple way to discover it.

In more detail, the main goal of this work is to find convenient data, preprocess them and build a sentiment classifier using neural network. It can recognize positive, neutral and negative emotions, and it is built especially for classifying tweets. Pre-trained model is used in a simple web application whatdoestwitterthink.com which demonstrates practical use of sentiment analysis and can be useful for individuals and businesses.

There are many existing solutions of sentiment analysis, both academic and commercial. Some of them are able to recognize wide spectrum of emotions like anger, frustration, anxiety, or happiness. Unfortunately, academic work is usually not accessible for public and commercial solutions are almost always paid. One good exception is Sentiment140[1] from Stanford University. It has an API for classifying tweets, but it does not have a nice user interface, therefore, it is useful primarily for developers. This is the added value of this work. Its result is useful for everyone, not only developers or researchers. Furthermore, the resulting application collects feedback from users. It can be eligible for further improvements and experiments.

[1] http://www.sentiment140.com/

2. Background

Social media, including Twitter, are one of the reasons why popularity of sentiment analysis increases [3] and approach to sentiment analysis is evolving.

2.1 Machine Learning

In the past, lexicon based approach, which determines emotion by polarity of the words, had been widely used.

Figure 1. This is the example of tweet, where simple lexicon based sentiment analysis fails. There are two positive words and only one negative, though overall sentiment should be negative.

Today, it is common to use machine learning and train classifiers with features such as unigrams or bigrams [4]. Algorithms like Naive Bayes, Maximum Entropy or Support Vector Machines can be also used. This work uses neural network.

When working with text, specific case of RNN (Recurrent Neural Networks), called LSTM (Long Short Time Memory) networks, is popular [5]. LSTM networks are able to remember context. Another suitable type are CNNs (Convolutional Neural Networks). This work experiments with both of them.

2.2 Natural Language Processing

Text has to be preprocessed, in order to get rid of unnecessary parts (for example URLs, user mentions in tweets). Preprocessing will be discused in section 3.2. After that, a dictionary is built, where every word receives its own index. Finally, the word embeddings are created.

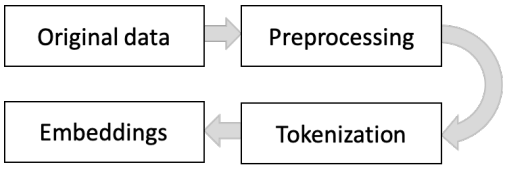


Figure 2. Text has to be preprocessed, transformed into tokens and embeddings.

Embeddings are able to formulate relationships between words. Even though deep learning libraries usually allow to use built-in embedding layer, it is common to use pre-trained models. One of them is Word2vec developed by Tomas Mikolov et al. [6]. Word2vec model trained on GoogleNews articles is used in this work[2].

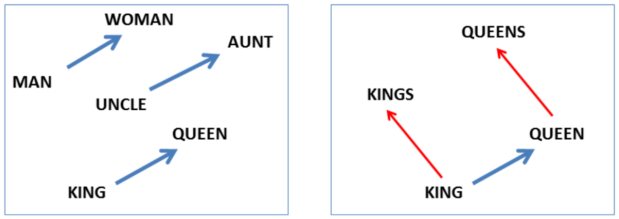


Figure 3. Word2vec can map relations between genders or singular and plural forms. Image taken from [6].

3. Data Gathering And Preprocessing

Tweets are very unique texts in many ways [7]. They are short, contain misspellings and slang.

3.1 Appropriate dataset

Initially, this work used data from aforemetioned project Sentiment140. It contains 1,6 milion tweets annotated as positive or negative according to occurrence of positive and negative emoticons. Number of positive and negative tweets is equal. Testing dataset, containing 500 manually annotated tweets, was also prepared within this project.

The tweets have been cleaned, which means they do not contain URLs, mentions of other users and emoticons. Missing emoticons appeared as a complication, which is demonstrated in figure 4. Moreover, there are only positive and negative tweets in training data, neutral class is missing.

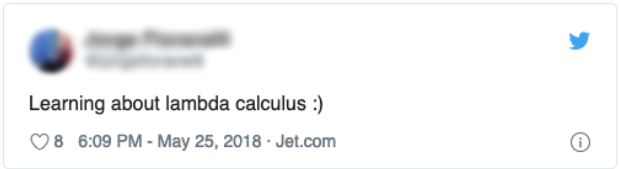


Figure 4. This tweet without emoticon would be considered as neutral, maybe negative for someone. But it is definitely positive with emoticon.

For that reason, additional datasets have been also used. It is quite simple to find a dataset consisting of tweets, but neutral emotion is almost always missing, although neutral class in training data is important [8].

An experimental approach was to download large dataset of tweets from Kaggle [3] without annotation and annotate them using Sentiment140 API. The results were good and most importantly, there was a neutral class.

[2] code.google.com – pre-trained Word2vec model
[3] Kaggle.com– Customer Support on Twitter

In the end, manually annotated data were added. Details about how different datasets influenced the results are listed in the section 4.

3.2 Data cleaning

Tweets are noisy. It is necessary to preprocess them, remove useless parts and try to normalize them. Usually, lemmatization and stemming are used to normalize words. Simply stated, stemming just chops off the end of words. Lemmatization is more powerful, therefore it is used in this work.

Original word	Stemming	Lemmatization
Caring	Car	Care
Stripes	Strip	Strip / Stripes

Table 1. Lemmatization considers the context. The problem of stemming is that more words can lead to the same form.

Common approach is to remove stop words, which do not contain any emotion (what, where, etc.). It can reduce the size of vocabulary, unfortunately, it slightly worsened the results, as it is shown in table 2.

After experimenting, the final preprocessing includes lowercase conversion, deleting user mentions, URLs, letters RT indicating retweet, some punctuation marks and transformating one or more occurence of character to two occurences. Thanks to that, "thaaaaanks" becomes "thaanks", which is something more positive than "thanks".

Figure 5. This is an example of preprocessing.

4. Implementation And Evaluation

In the beginning, LSTM newtork and 1,6 milion tweets from Sentiment140 were used for training and testing. Since only positive and negative tweets are present in training dataset, it was only binary classification. No pre-trained embeddings were used. For testing, there were two sets. The first one contained 1500 unseen tweets from original 1,6 milion. The second one contained only manually annotated tweets, also only positive and negative. Accuracy on both of these sets were almost the same – 80 % and it did not increase with any of the made experiments.

Training LSTM networks is computationally expensive, hence the experiments were limited to fewer architectures and hyper-parameters. This was the main reason why, all things considered, different type of neural network was used for another experiments. It was observed that the performance of CNN classifier is comparable to that of LSTM. Moreover, CNN models are computationally inexpensive.

CNN has an embedding layer, which uses pretrained Word2vec model. Sequences obtained from tokenizer are its input. It maps each word of a tweet to a feature vector and outputs matrix. Then, the network consists of five convolutional layers which capture contextual information. Each of them is followed by max-pooling layer, extracting the largest values and reducing size. After passing through these layers, dense and dropout layers are used. The last dense layer uses softmax activation function, which outputs the probability value for each class.

At this stage, it turned out that it is necessary to handle the neutral emotion. One of the possible solution was not adding neutral training data and considering those sentences, whose probability for positive and negative classes is similar as neutral. This approach was not successful, as it is illustrated in table 2.

Another idea was to change the dataset, experiment with representation of classes and sample weights. Different methods of handling emoticons were used. One of them was representing emoticons as text – for example "emojistart grinning face emojiend".

	1	2
Data	Positive and negative only, threshold for neutral set manually	Stop words removed
Acc. on validation dataset	81 %	78 %
Acc. on manually annoted dataset	58 %	59 %

Table 2. These table shows two experiments with poor results. The first one involved only positive and negative training data, although there was neutral class in testing dataset. The second one used stop words removing technique.

Several datasets were used for other experiments. Finally, the best results were obtained from classifier,

	1	2	3
Dataset	Customer Support on Twitter	Customer Support on Twitter + 1000 manually annotated	mixed + emoticons
Dataset details		same number of positive, negative, neutral tweets	emoticons in text replaced by “emojistart emojiname emojiend”
Acc. on validation dataset	86 %	85 %	80 %
Acc. on manually annotated dataset	54 %	62%	53 %

Table 3. This table compares different experiments. All of them used dataset mentioned in section 3.1 annotated with Sentiment140, the second and third ones added another data and used sample weights. The second one had the same number of samples from each class, the third one replaced emoticons.

which worked with dataset, whose sentiment was annotated with Sentiment140 API. Considering the fact that testing dataset is also from project Sentiment140, it is reasonable.

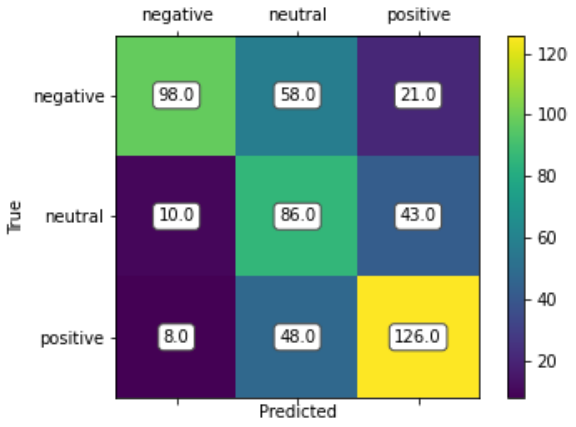


Figure 6. Confusion matrix (belongs to experiment 2 mentioned in table 3) shows that neutral class has the most false positives.

62 % seems not to be great, but this number alone actually does not say almost anything. It expresses how good the model is on this particular dataset from Sentiment140. The problem is, this dataset is from 2009 and it might be considered as slightly different from modern tweets. For example, modern tweets contain emoticons which did not exist 11 years ago. Therefore, further testing on different datasets is needed to improve the classifier.

Above that, the web application collects feedback from users and it is planned to use this data to improve the classifier. More about the method of collecting feedback can be found in section 5.

5. Web Application

Pre-trained model was used to create a web application whatdoestwitterthink.com. This application is based on client-server model.

Client is represented by a simple webpage implemented with HTML, CSS, and JavaScript. Python web server was made with Flask framework. Its task is to download data from Twitter, preprocess, and classify them. It uses pre-trained model and tokenizer.

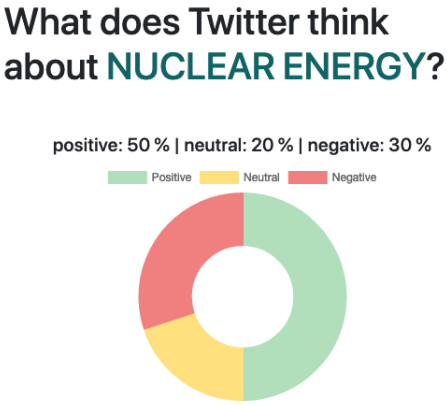


Figure 7. The main task of the web application is to classify how many tweets are positive, negative and neutral.

When a user writes down a topic, website sends a request to Flask server. It downloads data, classifies them, and sends a response in JSON format. Website processes this JSON and displays data as charts etc. It also shows examples of classified tweets.

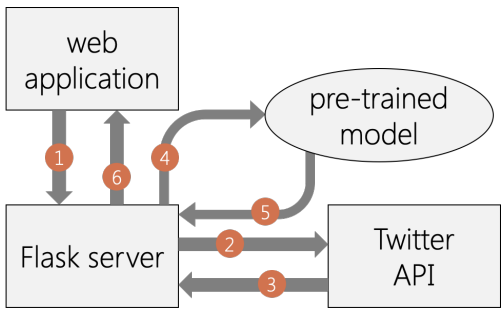


Figure 8. After user writes down a topic, the request to a Flask server is sent (1). Server sends request to Twitter API (2), API returns tweets in JSON format (3). Flask server uses pre-trained model to classify tweets (4, 5) and sends a response to web application in JSON format (6).

The Flask server uses SQLite database, therefore, the history of searches is saved. The app is able to display last searches and how were emotions changed during time.

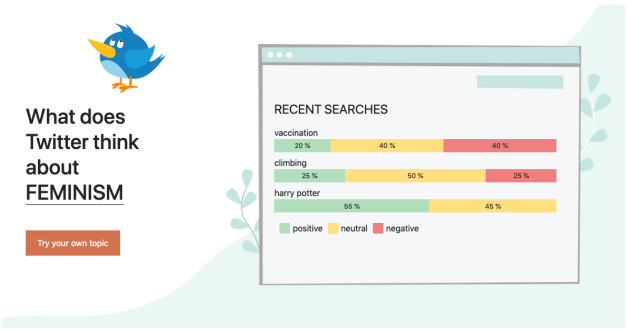


Figure 9. Since the web application uses SQLite[4], it is able to display recent searches.

The application shows examples of classified tweets; and users can correct them, if they are classified incorrectly. Corrections are saved in a database for further improvements of classifier.

6. Conclusions

This paper describes one of the possible approaches to sentiment analysis – training neural network. It also includes basic techniques of text preprocessing.

Pre-trained model is then used in web application whatdoestwitterthink.com, which allows its users to analyze sentiment from tweets and collects feedback from users. Target group of this application can be anyone, since it was made for the general public.

There are many opportunities for improvement. First of all, it is necessary to test classifier on different datasets. Additionally, the classifier is not able to cope with sarcasm, tweets containing both positive and negative emotions etc.. Moreover, there are tweets in which it seems impossible to decide whether they are positive or negative.

Figure 10. The user agrees (positive). But he is also angry where were American values (negative).

The main goal right now is to collect feedback from users and use it for additional improvements.

[4] SQLite.org – C-language library that implements a SQL database engine

Acknowledgements

I would like to thank my supervisor Ing. Igor Szőke, Ph.D. for his help and assistance, Ing. Ondřej Novotný for his advices and provided data and Ing. Sebastián Poliak for many ideas what to try and how to improve my work.

References

[1] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. *Proceedings of the Workshop on Languages in Social Media*, 01 2011.

[2] Walter Kasper and Mihaela Vela. Sentiment analysis for hotel reviews. *Speech Technology*, 4:96–109, 11 2012.

[3] Bing Liu and Lei Zhang. *A Survey of Opinion Mining and Sentiment Analysis*, pages 415–463. Springer US, Boston, MA, 2012.

[4] Lei Zhang, Riddhiman Ghosh, Mohamed Dekhil, Meichun Hsu, and Bing Liu. Combining lexicon-based and learning-based methods for twitter sentiment analysis. 01 2011.

[5] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. A C-LSTM neural network for text classification. *CoRR*, abs/1511.08630, 2015.

[6] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[7] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, 150, 01 2009.

[8] Moshe Koppel and Jonathan Schler. The importance of neutral examples for learning sentiment. *Computational Intelligence*, 22(2):100–109, 2006.