Exploring Different Methods for Tweet Sentiment Classification

Charlotte Dalenbrook
I6155056
Maastricht University
Faculty of Science and Engineering

Arnaud Ruymaekers
I6151654
Masstricht University
Faculty of Science and Engineering

Abstract

This paper compares several methods to classify the sentiment of a set of 1.6 million tweets from the Sentiment140 data set. We implement a bag of words model with a naive Bayes classifier, an N-grams model with a naive Bayes classifier, and a combination of a convolutional neural network (CNN) and an long short-term memory (LSTM) recurrent neural network. We compare the methods by contrasting the accuracies we can achieve in the classification of a test set using them.

1 Introduction

With the rise of social media such as Facebook, Twitter and others, huge amounts of data have become available for companies to use for various purposes [1]. Sentiment classification, among others, is a method that can be used to extract the useful data from the large data sets [1]. The interest in sentiment classification of data sets such as tweets, among others, has therefore risen significantly in the past years [1] [2] [3] [4].

This paper aims to survey several well known techniques for sentiment classification and analyse their performance. We use a well known data set "Sentiment140", which contains 1.6 million tweets that are classified according to their sentiment [5]. The training data was labelled automatically through the assumption that tweets containing positive emojis such as ":)" were positive, and tweets containing negative emojis such as ":(" are negative [5]. Additionally, emojis were removed from the tweets and tweets containing both positive and negative emojis as well as retweets were removed from the data set [5].

The techniques that we implemented in order to analyse included Bag of Words, N-Grams and a combined CNN-LSTM. These were chosen as they are some of the most popular and well known methods used in the area of natural language processing. In order to compare their performance we compare the accuracy's they achieve. Additionally, we explored how different kinds of preprocessing and changing other parameters affected the accuracy of the models.

2 Related Work

Sentiment analysis in general is a hot topic in research at the moment, with implications reaching from helping authorities in properly allocating resources for disaster relief [6] to classifying tweets according to topics in order to analyse them [7]. Due to these factors a lot of work has already been done on the sentiment analysis of tweets, e.g. [3], [5], [8].

Recurrent Neural Networks (RNN) [9] as well as Gated Recurrent Units (GRU) [10] are both popular methods implemented in order to tackle the classification of short pieces of text. Long Short Term Memory (LSTM) networks have also been used extensively when classifying sentiment since they avoid the diminishing gradient problem which normal RNNs encounter [1]. Another method that has been extensively used for short text classification are Convolutional Neural Networks (CNN) [11].

As proposed by Sosa [1], a Convolutional Neural Network (CNN) and Long Short-Term Memory Neural Network (LSTM) can be combined in order to further improve over the accuracy of tweet classification using a CNN or LSTM alone. In his paper Sosa showed that the average accuracy of a combined LSTM-CNN network performs better than both the CNN and the LSTM alone [1].

3 Methods

In order to compare which method was the most accurate for tweet sentiment classification, we im-

plemented 3 different approaches. We also made sure to pre-process the text so that the input was as clean as possible.

3.1 N-Gram

Unlike the bag-of-words model, an n-gram model takes into account the context of the words in a text. Probabilities of words or phrases occurring similarly to the bag-of-words model, by summing their occurrences in a text and dividing by the total number of tokens. In our case we made a set of n-grams for tweets classified as good, as well as a set of n-grams for tweets classified as bad. We then computed the probability of a sentence being good or bad by summing the log probability of each word of the sentence being in the class and comparing it with the same measure in the other class.

When calculating the probabilities of the n-grams, we used Stupid Backoff alongside Laplace smoothing in order to make sure that never seen n-grams aren't assigned a probability of zero. Stupid backoff on the other hand tries first the n-gram, if the count of this is zero, it will try the (n-1)-gram. It does so until it reaches the unigram level, there, Laplace smoothing will be applied in order to avoid zero-probabilities. Once the probabilities of the n-grams were calculated, these were used to predict the sentiment on tweets in the test set. A phrase such as "super excited" is an example of a bigram that has a very high probability of score, and therefore increases the score of the sentence being tested significantly in the positive direction.

For testing the accuracies of using n-grams with n from 1 to 5 alonngside Stupid Backoff will be tested.

3.2 Bag-of-Words Model

A bag-of-words model is a method of modelling a text through the counting of the words within it, disregarding any structure or order. The bag-of-words model assumes that two texts are similar if they contain similar words. In the context of tweet sentiment classification, if a tweet contains a similar vocabulary to a previously positively classified tweet, it is also considered to be positive.

We implement a bag-of-words model for each class on our data set and classified the tweets as positive or negative using a naive Bayes classifier.

3.3 LSTM

LSTM networks are a kind of recurrent neural network that can remember previously read values [1]. An LSTM solves the diminishing gradient problem that regular recurrent neural networks experience through units within the network that essentially decide which information is important to keep or remember [12]. An LSTM achieves this through so called memory cells within the network which are made of an input gate, a forget gate, and an output gate [12].

Our implementation of an LSTM is made of 3 layers. The first layer makes the embeddings. We turn every word in all tweets into an index. Each tweet is then represented as a list of indices. These indices can then be turned into a dense vector using Keras. The second layer is the actual LSTM and the final one is a dense neural network classifier.

3.4 Pre-Processing

To make sure that the models only received input relevant to the analysis of the tweets, several steps were taken to pre-process the text. Firstly, all emojis were replaced with the generic phrases. "emojihappy" was used for positive emojis and "emojisad" for negative ones. Next the @ symbol for mentions of other users within tweets was removed and replaced with the word "mentiontoken". URL's were also removed and replaced with "urltoken". Finally some general clean up was done and all words were made lowercase and the remaining sentences were tokenized.

4 Approach

In order to compare the three implemented methods to one another we compared their accuracy in classifying a test set after training.

4.1 Experimental Setup

Each of the 3 methods was run for 10 epochs.

Additionally, tests were made to explore the effect of different preprocessing methods on the results. Firstly, we tested the effect of removing stop words from the tweets on the accuracy. Next, we removed all repeated characters in words in order to turn occurances of words such as "ohhhhh" into their base form (in this case "oh"). Finally we changed the phrase that replaced different emojis from a different phrase for each emoji to simple a "happyemoji" phrase for all emojis considered

Model	Accuracy
Bag-of-Words	77.2338%
Unigram	78.1255%
LSTM [1]	72.5%

Table 1: Accuracies of the 3 Models Implemented

Discount Factor	0.4	1
Unigram	0.781255	0.781255
Bigram	0.765300	0.7694825
Trigram	0.722160	0.724405
Quadgram	0.685925	0.683025
Pentagram	0.65341	0.6479975

Table 2: Accuracies of Uni to Pentagrams with Discount Factors

positive, and the "sademoji" phrase for all emojis considered negative. In order to test all effects accuracies were compared to those achieved when only base line preprocessing (as shown in section 3.4) was applied to the tweets.

5 Results

5.1 Model Accuracies

The accuracies achieved by the various models are shown in table 1. The model that achieved the best accuracy was . Next was . The worst performing model with regards to accuracy was with accuracy.

Table 2 shows the accuracies obtained with different n-grams as well as different discount factors. Unigrams performed the best overall for both discount factors. Bigrams and trigrams had a better accuracy with a discount factor of 1, while the accuracy for the quadgram and pentragram was better with a discount factor of 0.4.

5.2 Pre-Processing Changes

With the baseline pre-processing mentioned in section 3.4, the unigram model achieved an accuracy of 78.13%. When removing stop words in the tweets the accuracy decreased to 77.65%.

Removing the stop words improved the accuracy for the bag-of-words model. With the stop words the accuracy of the model was 76.0000%, while without the stop words in the tweets the accuracy was 77.23338%.

Changing the phrase that emojis were replaced with from a unique phrase per emoji to two phrases, "happyemoji" and "sademoji", caused

Model	With Stop-words	Without Stop-words
Unigram model	0.78254	0.776525
Bag-of-Words	0.76	0.7723375

Table 3: Accuracies of Stop-words removal at preprocessing on different models

Model	2 Emoji Tags	Unique Emoji Tags
Unigram model	0.78254	0.776525
Bag-of-Words	0.771675	0.772338

Table 4: Accuracies of merge of the emoji tags on different models

the accuracy of the unigram to increase from 77.6525% to 78.2540% with the rest of the preprocessing being like in section 3.4. For the bag of words when making two phrases the accuracy decreased from 77.2338% to 77.1675%.

6 Analysis

As can be seen in table 1, the unigram achieved the highest accuracy of all the models with 78.13%. It is especially remarkable that unigrams significantly outperforms any other n-gram, suggesting that in the case of tweets, context is not necessary to accurately predict the sentiment of a tweet. This is most likely due to the length restrictions of tweets, meaning users are not able to give much context anyways. Additionally, tweets are very informal, and don't have to follow any gramatical rules or format, meaning lower chances of the same words being in an n-gram and therefore making the method less useful in predicting sentiment. Additionally, it should be noted that the approach of an LSTM as done by Sosa [1], achieved an accuracy of only 72.5%. This suggests that tweets can be quite easily classified with simple methods, and don't require more complicated neural networks such as an LSTM.

It is surprising that the accuracy decreased for the unigram model when removing stop words from the corpus. This could be attributed to a factor such as stop words being used more often in either a negative or positive context. For example, if frustrated, a user may use more unecessary filler words. This could mean that stop words actually do have a correlation with the sentiment of a tweet and therefore removing them causes the unigram to perform worse. This is unlikely however, since

for the bag-of-words model the accuracy was improved by removing stop words from the corpus.

Changing the phrase that emojis were replaced with also had different effects for the two methods. For the unigram approach, classifying emojis as only 2 different types, happy or sad, made the accuracy increase slightly. For the BOW model on the other hand, the accuracy slightly decreased due to this change.

Overall, the way the different models reacted differently to the same changes in pre-processing shows that pre-processing should be adjusted depending on what model is being implemented.

7 Future Work

Due to time and computation constraints for this project many interesting aspects could not be explored. It would have been interesting to compare several more methods to the ones already implemented. Furthermore, it would be interesting to explore how more methods of pre-processing affect the performance of all models. Finally, changing and perfecting the parameters for each model could improve results further, and possibly allow a model to become better than another.

8 Conclusion

In this paper, several methods of tweet sentiment classification were presented and analysed. It was shown that even a very simple unigram model can give very high values for classification, and not much context is needed to accurately predict the sentiment of tweets. Additionally, it was shown that depending on the model used, different types of pre-processing are necessary.

References

[1] Pedro M. Sosa. Twitter sentiment analysis using combined lstm-cnn models. Unpublished paper, 2017.

[2] Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 208–212, 2014.

[3] Vishal Kharde, Prof Sonawane, et al. Sentiment analysis of twitter data: a survey of techniques. *arXiv preprint arXiv:1601.06971*, 2016.

[4] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1555–1565, 2014.

[5] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009, 2009.

[6] Zahra Ashktorab, Christopher Brown, Manojit Nandi, and Aron Culotta. Tweedr: Mining twitter to inform disaster response. In *ISCRAM*, 2014.

[7] Kathy Lee, Diana Palsetia, Ramanathan Narayanan, Md Mostofa Ali Patwary, Ankit Agrawal, and Alok Choudhary. Twitter trending topic classification. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 251–258. IEEE, 2011.

[8] Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. Twitter sentiment analysis: The good the bad and the omg! In *Fifth International AAAI conference on weblogs and social media*, 2011.

[9] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[11] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[12] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.