KTH Royal Institute Of Technology

Big Data Sentiment Analysis in Social Media

Big Data In Media Technology, DM2583

Eysteinn Gunnlaugsson
eysgun@kth.se

Carlo Rapisada
carlora@kth.se

Steinn Elliði Pétursson
petu@kth.se

Yunus Koçyiğit
yunus@kth.se

July 18, 2018

1 Abstract

Sentiment analysis has been used with good results to analyze peoples' opinions on different topics. This can provide valuable insights for companies and individuals by seeing how their actions can affect their reputation. In this paper we explore different approaches to sentiment analysis in social media texts, show results from applying them on live data from social media and how unstructured data can be analyzed to provide insights into the opinions of the hive mind.

2 Introduction

In today's world social media has become a large part of society. People express their opinions on the internet every day through sources like Twitter and Facebook. The data posted contains information about people's opinions on day to day events. Recent studies have shown that there are 500 million tweets posted every day worldwide[1]. This paper will focus on the task of text categorization and sentiment analysis on data available online. Specifically, with the recent advances in big data research, is it possible to determine the views of the hive mind on known entities in the world through twitter by comparing sentimental values of people before and after controversial events.

3 Data Gathering

For this project three different sources were utilized for training, validation and testing. The VADER dataset [10] provided valuable data for training the classifiers for twitter data classification. The SNAP dataset [12] consists of nearly 8 million movie reviews which were used for training and validation of the classifiers. Finally the twitter API was used to gather unlabeled tweets for classification and analysis of live data.

3.1 Amazon Reviews Dataset

The SNAP dataset was used to train the Neural Network Classifiers. The dataset consists of 7.911.684 reviews on 253.059 different products from 889.176 different users. For this research only 1.250.000 reviews were used. These reviews were randomly selected from the entire dataset. 200.000 for training, 50.000 for validation and 1.000.000 for testing. The review text and 1-5 star rating was extracted. The 1-5 star rating was converted to three categories. 1-2 stars was considered negative, 3 stars was considered neutral and 4-5 stars was considered positive.

3.2 Vader Dataset

The VADER dataset and sentiment analysis tool is an open source sentiment analysis tool which provides a dataset of 4000 tweets rated by 20 independent human raters which makes it an excellent dataset for training social media classifiers.

3.3 Twitter Dataset

Since one of the goals was to obtain insights from social media, an important component of the project architecture is the one related to the gathering of the data posted by users. Twitter is the perfect source for the project purposes, as the entire social network revolves around mostly text-based entries. In addition, Twitter provides comprehensive APIs accessible by researchers and developers around the world, together with powerful search and filtering options, therefore there was no need to implement a data-scraping algorithm.

3.3.1 The twitter search API

Twitter offers three different options with different capabilities: a standard option, and two enterprise options. Given the researchers limited resources, it was decided

to take advantage of the free standard package, which gives access to data from the past 7 days, limited to 450 queries (retrieving max 100 tweets each) within a 15 minutes window; this means that it is possible to gather up to 180,000 tweets per hour (ignoring connection delays). For the purposes of this project, it was decided that these capabilities were sufficient, although access to historical data to analyze trends over a longer time frame would have been preferable.

3.3.2 Data gathering methods

Each request to the Twitter API returns up to 100 tweets, therefore a Python script was written to execute as many requests as needed to gather the wanted number of entries. The unique identifier associated with each tweet allowed the researchers to retrieve paged results. In addition, the Twitter API had the option to obtain recent, popular, or mixed posts; in this project it was decided to gather the recent tweets, as the researchers were interested in the opinions of all users, rather than those of public figures.

After gathering the data with the official API, URLs were stripped from the texts, and retweets converted into regular tweets to have a consistent format for all entries. The data was then written to a file in a CSV format, ready to be used by other components of the project architecture.

4 Pre-processing Methods

Sentiment analysis is the procedure of extracting the traction a writer has towards some object, being it negative, positive or somewhere in between. The idea of using sentiment analysis to classify future online posts is something that has been the topic of research for some time now. Since 2004 there have been nearly 7000 papers published on the topic[2]. Due to the extensive knowledge base existing on this subject the researchers decided to focus their efforts on a few known and tested text pre-processing methods outlined in the next chapters.

4.1 Filtering

Twitter data can often contain a lot of noise which has no value for a classifier such as links and non-English characters such as emojis. The filtering process used focused on taking out such noise in the data. In this research the target language was English so non-English characters were filtered out. Additionally in the twitter data gathered all URLs were stripped from the data used. Another filter tried in this project but not used in the final product was the filtering of known stop words in the English language. In longer documents filtering out stopwords is essential to reduce the feature space and removing noise from the data. The researchers found that in the case of Twitter sentiment analysis this proved harmful for the classifiers often decreasing their performance by a significant margin. This is consistent with the findings of other researchers in the field [3].

4.2 Word Stemming

Another method utilized was the process of word stemming [4] which reduces words down to their word stem, for example the word 'government' will be reduced to the word stem 'govern'. This method is used to reduce the feature space even further than filtering manages to do. In this project the NLTK Lancaster stemmer [5] was utilized for word stemming.

4.3 Text Vectorization

The last step in the text preprocessing was the conversion of documents to value based vectors from a constructed vocabulary of

words. The vocabulary and word mapping was constructed using the tf-idf algorithm, which assigns values to words in document based on their importance relative to the document and the whole corpus.

> *...TF-IDF works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus. Intuitively, this calculation determines how relevant a given word is in a particular document. Words that are common in a single or a small group of documents tend to have higher TF-IDF numbers than common words such as articles and prepositions.*
> Juan Ramos. (2003). Using TF-IDF to Determine Word Relevance in Document Queries. 2, 19-27.

After running this algorithm on the training data, the final step was normalizing the values of each document vector to have a length of 1, using the sklearn vector normalization package with l2 normalization [6]. This was done to eliminate the difference document length made on document importance. The tf-idf algorithm gives longer documents higher importance in the form of higher vector values for longer documents which proved not essential in this project.

5 Classifiers

The classifiers implemented were a Naive Bayes Classifier(NBC), Support-Vector-Machine(SVM), Feed-forward Neural network Classifier(FFNN) and Convolutional Neural network Classifier(CNN). Through the years a lot of research has been done on sentiment analysis using a NBC or a SVM with good results and these methods have been the gold standard for a while [7]. FFNN and CNN are however quite new and seem to be surpassing the golden standard methods [8].
Our hypothesis is that the more current classifiers, FFNN and CNN, will outperform both the NBC and SVM. However we believe that the amount of data to get a robust NN classifier will be far more than the data needed for the old methods. We will therefore try these two schools of classifiers on two seperate sources of data.

5.1 Classifier Evaluation

For the training of each classifier the data set was split in two parts, 80% training, 20% validation. The performance was then evaluated by looking at the F1-score [21] for the validation set. Confusion matrices and classification reports were used to visualize results in a descriptive way.

5.2 Naïve-Bayes Classifier

The NBC classifier is relatively simple compared to the other classifiers so little hyperparameter tuning could be made to improve validation accuracy. The results from applying the classifier on the VADER data set can be found in appendix A. An overall F1-score of 0.62 was reached which is pretty good for such a simple algorithm. The main benefit from using a NBC is that the model obtained is not a black-box model and can therefore be very descriptive.

5.3 Support-Vector-Machine

A SVM classifier was also implemented for analyzing the data and comparing the results. In this project the SVM classifier brings better results for the current data. It does not mean that SVM is better than NBC, rather it depends on the data [20] used in each case for NBC and SVM. The SVM reached a F1-score of 0.7 which is very

good since the SVM is very simple and can be implemented with ease

5.4 Neural Network Classifiers

With the help of the Keras python library it is very easy to get a neural network up and running. It is however a tedious task to optimize all the hyper-parameters that can be tuned when building a neural network. A lot of effort went in to finding the optimal values for these parameters. To address the issue of over-fitting to our training set a technique called dropout was used [9] with good result. At first we trained our networks on the VADER dataset. The results from that were not so different than the results using a SVM. It was our hypothesis that more data was needed to build a neural network classifier to see different results. For these reasons both neural networks used 250000 reviews from the Amazon data set, 200000 for training 50000 for validation. The output from the models was the likelihood of each class(negative, neutral or positive), to evaluate the classification performance the class with the highest likelihood was chosen as the class predicted.

5.4.1 Feed Forward

The final version of the FFNN had 4 hidden layers with 256, 128, 128, 64 nodes respectively. For the hidden layers RElU activation functions were used. Each hidden layer had a Dropout chance of 0.6.
The results from applying the classifier on the Amazon dataset can be found in appendix A:. A F1-score of 0.7 was reached on the validation set which is pretty good given that the network architecture is very simple.

5.4.2 Convolutional

The version of the CCNN had 4 hidden layers. Two of them were convolutional layers and after them two dense layers were added. The first convolution layer had 10 filters and a kernel size of 4. The second convolutional layer had 10 filters and a kernel size of 8. Both of the dense layers had 128 nodes. All layers used RElU activation functions. The idea for this model architecture was derived from a very successful model by Alexander Rakhlin [17]. The results from applying the classifier on the Amazon dataset can be found in appendix A. A F1-score of 0.72 was reached, which is the highest we saw for any of our classifiers. This is very promising and it is our belief that this score can be improved with longer training time, more data and minor changes to the structure.

6 Programming Frameworks

The programming language used in this project was the Python programming language. Various libraries were used to assist the researchers in their work throughout the project. NLTK [13] provides a Lancaster stemmer as well as an extensive stop word corpus for the English language. Scikit-learn [14] has available a tf-idf vectorizer which was used in the data preprocessing for the classifiers. Scikit-learn also offers implementations of Naïve-Bayes classifiers and Support vector machines which were used in the project. Finally, Scikit-learn has a good implementation for visualization of confusion matrices and matrix to measure a models performance which was used to generate the tables in appendix A. For both the convolutional neural network classifier and the feed forward neural network classifier the Keras deep learning python library was utilised [15]. Keras is a library built on top of the Tensorflow framework [16]. Since Keras utilises the Tensorflow neural network the researchers set up tensorflow with GPU support to speed up the neural net optimisation with an NVIDIA GPU, this resulted in major speed upgrades in training the neural nets compared to using the CPU for training.

7 Results

In this project four different classifiers were used with good results. As expected the NBC classifier had the least success with this data with a F1-score of 0.62. NBC classifiers are relatively simple compared to the other classifiers and are more effective on more distinctive datasets. The other three classifiers all performed significantly better, the SVM and FFNN with an F1-score of 0.70 and the CNN with an F1-score of 0.72. This shows that an SVM classifier is still a very relevant classifier with the given data. The CNN classifier performed better than the others and was still showing improvements in the end. We believe that with better tuning and more data the classifier could still be improved significantly. It was apparent in the research that the neural net classifiers needed significantly more data to function as well as the other classifiers and with even more data ($>$ 1.000.000 entries preferably) they could still make significant improvements. As the CNN classifier had the best results on the validation dataset (50.000 entries) this classifier was tested on 1.000.000 entries from the SNAP amazon dataset, the results from that experiment can be seen in Appendix B. The testing went better than expected, on the 1.000.000 entries the classifier managed an impressive F1-score of 0.80. All of the classifiers seemed to struggle with classifying neutral data in the sentiment analysis, in the test dataset the entries were not uniformly distributed as the dataset did not contain enough neutral entries. Therefore the test dataset had relatively more negative and positive entries than the validation dataset. This can partly explain the significant difference in performance between the test dataset and the validation dataset. As a proof of concept for the work done in this project a simple demo [19] was implemented on the live twitter data gathered. This demo shows the positive and negative tweets related to a particular hash tag over a timeframe, as well as the most informative features in the tweets. An example of this work with analysis of 500.000 tweets gathered from the hashtag "#trump" can be found here[22].

8 Future work

There are countless ways to set up a text preprocessor. In this project some of the most known and battle tested ones were used. Other methods, like synonym transformation where synonyms are normalized. For example great and excellent have similar meanings so the word excellent would always be substituted for the word great. This method can help with reducing the feature space for the classifiers without it impacting the classifier performance. Vader provides a good sentiment lexicon [11; 10] which could be incorporated into the preprocessor vocabulary by adding sentiment values to words before they are classified. Another well known method is Part-Of-Speech-tagging [18] which could further improve the classifier if utilised wisely. It was apparent in the research that all the classifiers struggled with classifying neutral sentiments, this was to be expected as the neutral sentiment had a narrower sentiment value from the SNAP data than the other two classifications. Adjusting for this in the data by making the neutral classification more broad at the cost of making the negative and positive data classification narrower would probably produce better results. The dataset used was aimed at classification in sentiment analysis. Getting a dataset similar to the VADER dataset which is set up as a regression model would likely produce more accurate results. This project showed promising results in terms of sentiment analysis on social media, we would like to take this research further and research how events in the real world affect companies and peoples' views on them.

References

[1] Salman Aslam,
Twitter by the Numbers: Stats, Demographics & Fun Facts, 2017,
https://www.omnicoreagency.com/twitter-statistics/

[2] Mika V. Mäntylä Daniel Graziotin2, Miikka Kuutila,
The Evolution of Sentiment Analysis - A Review of Research Topics, Venues, and Top Cited Papers, 2016

[3] Hassan Saif, Miriam Fernandez, Yulan He, Harith Alani *On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter*, 2014

[4] Jasmeet Singh, Vishal Gupta, *Text Stemming: Approaches, Applications, and Challenges*, 2016

[5] Nltk source code for the Lancaster Stemmer,
http://www.nltk.org/_modules/nltk/stem/lancaster.html

[6] Vector normalization in sklearn
http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html

[7] Pyry Takala, Pekka Malo, Ankur Sinha, Oskar Ahlgren,
Gold-standard for Topic-specific Sentiment Analysis of Economic Texts, 2014

[8] Yoon Kim,
Convolutional Neural Networks for Sentence Classification, 2014

[9] Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, 2014

[10] The VADER sentiment analysis tool and dataset
https://github.com/cjhutto/vaderSentiment

[11] C.J. Hutto, Eric Gilbert,
VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text, 2014, 2

[12] The SNAP amazon dataset
https://snap.stanford.edu/data/web-Amazon.html

[13] NLTK, natural language toolkit
http://www.nltk.org

[14] Scikit-learn, machine learning library in python
http://scikit-learn.org/stable/

[15] Keras, The Python deep learning library
https://keras.io

[16] Tensorflow, An open-source software library for Machine Intelligence
https://www.tensorflow.org

[17] Rakhlin, Alexander. *"Convolutional Neural Networks for Sentence Classification in Keras"*
`https://github.com/alexander-rakhlin/CNN-for-Sentence-Classification-in-Keras`

[18] Ratnaparkhi, Adwait.
"A maximum entropy model for part-of-speech tagging." Proceedings of the conference on empirical methods in natural language processing. Vol. 1. 1996.

[19] Link to the code used to implement the tests, as well as a live twitter demo utilising the classifiers on live twitter data
`https://github.com/steinnp/Big-Data-Final`

[20] Wang, Sida, and Christopher D. Manning.
"Baselines and bigrams: Simple, good sentiment and topic classification."
Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2. Association for Computational Linguistics, 2012.

[21] F1 - score, used to determine classifier accuracy
`http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html`

[22] Power BI Dashboards - Example of how data can be visualized
link

Appendix A - Validation Classification Results

Naïve Bayes Classifier Results

	Negative	Neutral	Positive	Precision	Recall	F1-Score	Support
Negative	101	17	106	0.81	0.45	0.58	224
Neutral	15	22	122	0.42	0.14	0.21	159
Positive	9	13	435	0.66	0.95	0.78	457
Average / Total				0.65	0.66	0.62	840

SVM Results

	Negative	Neutral	Positive	Precision	Recall	F1-Score	Support
Negative	140	26	59	0.74	0.62	0.68	225
Neutral	21	59	70	0.48	0.39	0.43	150
Positive	27	39	399	0.76	0.86	0.80	465
Average / Total				0.70	0.71	0.70	840

FFNN Results

	Negative	Neutral	Positive	Precision	Recall	F1-Score	Support
Negative	15636	2274	2166	0.78	0.78	0.78	20076
Neutral	2572	4235	3186	0.47	0.42	0.44	9993
Positive	1900	2565	15466	0.74	0.78	0.76	19931
Average / Total				0.70	0.71	0.70	50000

CNN Results

	Negative	Neutral	Positive	Precision	Recall	F1-Score	Support
Negative	16371	1336	2255	0.79	0.82	0.80	19962
Neutral	2940	3292	3836	0.56	0.33	0.41	10068
Positive	1521	1212	17237	0.74	0.86	0.80	19970
Average / Total				0.72	0.74	0.72	50000

Appendix B - Test Classification Results

CNN Results

	Negative	Neutral	Positive	Precision	Recall	F1-Score	Support
Negative	114532	9768	11834	0.55	0.84	0.66	136134
Neutral	30357	37149	31228	0.34	0.38	0.36	98734
Positive	64272	63492	637368	0.94	0.83	0.88	765132
Average / Total				0.82	0.79	0.80	1000000