

Detecting sentiment, humor, and truth: Sentiment Analysis in Twitter

Boštjan Bohte, Nikola Balaban

Faculty of Computer and Information Science, University of Ljubljana.
Večna pot 113, 1001 Ljubljana, Slovenia
<http://www.fri.uni-lj.si/>

Abstract. The basis of this project was to create a program which can predict the sentiment of a message posted on Tweeter by using common NLP tools and techniques. Idea was to see the impact of word frequency, emoticons and other word sentiment to sentiment of the whole message. We first processed text with lemmatisation and tried three techniques on that data. The outcomes were very different but we could see that with small changes in data processing we get significant improvements.

Keywords: NLP, classification, SemEval 2017, sentiment analysis, random forest classifier , TD-IDFT

1 Introduction

Social media has greatly democratized content creation. Facebook, Twitter, Skype, Whatsapp and LiveJournal are commonly used to share any thoughts and opinions about anything in the surrounding world. All this content has created new opportunities to study public opinion. Twitter is especially popular for research due to its scale, representative, variety of topics discussed and its ease access to content. In the past years, research in that direction was hindered by the unavailability of suitable datasets and lexicons for system training, development and testing. Some Twitter-specific resources were developed, but they were either small and proprietary, or they relied on noisy labels obtained automatically. This all changed with the shared task on Sentiment Analysis on Twitter, which is part of the international Worksop on Semantic Evaluation. The task is active since 2013 and it attract over 40+ participant teams. It contains 5 subtasks with their train and test datasets. First task is Message Polarity Classification, where we need to classify whether the given message is of positive, negative, or neutral sentiment. In second and third task, there is beside a given message, also a topic and you need to classify the message on two point scale and five-point scale, depending on the task. Forth and fifth task are also on two point and five point scale, where there are given tweets about a given topic, and you need to estimate their distribution [4].

2 Dataset

For tasks we are provided with their datasets with annotated tweets. They are gathered in a way, that express sentiment about popular topics. For this purpose, they extracted named entities from millions of tweets. The collected tweets were greatly skewed towards the neutral class.

To reduce the class imbalance, they removed those that contained no sentiment-bearing words. Tweets are then manually filtered to obtain a set of meaningful topics with at least 100 tweets each. Topics that are ambiguous (e.q., Barcelona, which is a city or sport club) or too general (e.q., Paris). The topics in the training and in the test data do not overlap, meaning that test tweets consist of topics that are different from the topics in train dataset. Dataset is consisted of four parts: TRAIN (for training models), DEV (for tuning models), DEVTEST (for development-time evaluation) and TEST (for official evaluation). The first three datasets were annotated using Amazon’s Mechanical Turk , while the TEST dataset was annotated on CrowdFlower [4].

3 Data Pre-Processing

First we tokenized and lemmatised the data and calculated frequency distributions of all data and divided data by sentiment. (*See Fig.1 for most common words*).

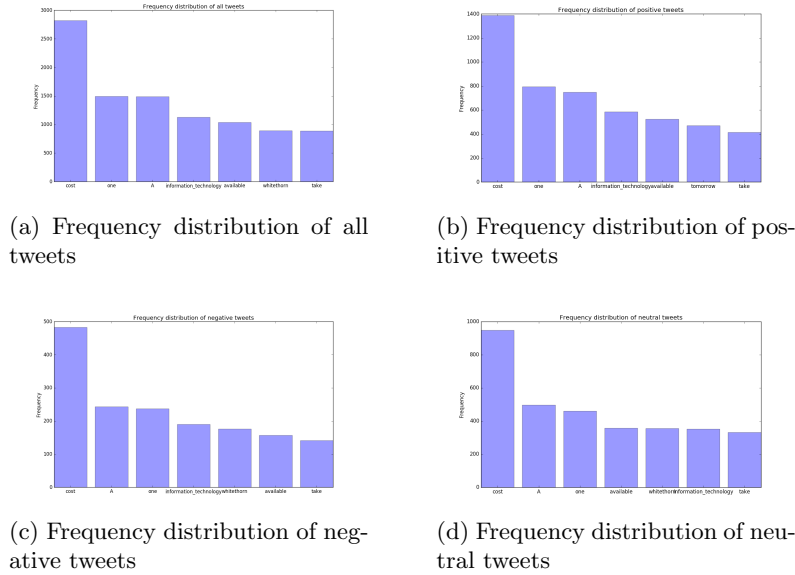


Fig. 1: Frequency distribution in data divided by sentiment

As we can see most frequent words and distribution is the same across all dataset because they have the same topic.

Then we filtered the profanity words to see if they have any impact on sentiment of a Tweeter message. *Fig.2* shows the frequency of profanity words by sentiment of Tweeter messages.

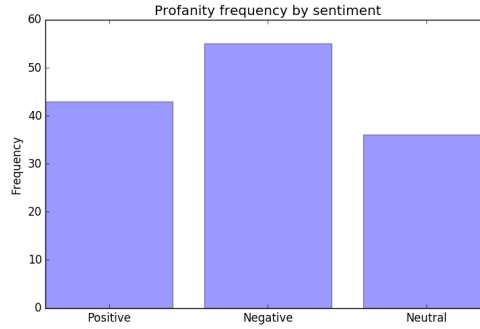


Fig.2: Frequency distribution of profanity words across Tweeter messages by sentiment

Both positive and negative sentiment tweets as we can see have more profanity words than neutral tweets. But still the use of profanity words is much common in messages with negative sentiment. This kind of data we can later use to optimize our weights for machine learning or regression or other classification methods.

Next we wanted to see the effect of emoticons on sentiment of the message. We tried one classification with three different datasets. One with no emoticons (1.), one with just some identifier for emoticon (2.) and one with sentiment of the emoticon included (3.). The tests were as following:

1. Classification accuracy: 38.93 %
2. Classification accuracy: 38.93 %
3. Classification accuracy: 38.98 %.

So we can see that the emoticons have some small impact on sentiment but it can be used even more to give more value to weights.

4 Related work

There are a lot of related work, that also competed in Sentiment Analysis in Twitter.

In work [1], they have trained convolutional neural network (CNN) to extract hidden activation values from the hidden layers once some input had been fed to

the network. That values served as features for SVM model. They have discovered that CNN serves good as feature extractor but not as good as independent classifier. They have received good results and were on the top of the rankings list. Their best results were for tasks B (2./14), C(4/11) and D(2/14). Their approach is very different from ours since they entirely use models with machine learning and our method is more calculating similarities between tweets.

In work [2] they have used simple unigram model baseline with three main features enchantments incorporated into the model. This features are emoticon retention, word stemming and token saliency calculation. This work is similar to ours, since they also uses TF-IDF to calculate similarities between tweets. They preprocessed text with word stemming, while we used lemmatizing. With stemming or lemmatizing, the emoticons get lost. In order to preserve emoticons, they have extracted it before stemming and use them as features. They also used unigrams to achieve better results. We didn't used them because we got worst performance.

5 Methods

5.1 Subtask B and Subtask C

For this tasks we have created a method, that first preprocess all train tweets and then sums all the tweets according to their belonging sentiment. With this approach we get two documents, one containing all the text for positive tweets and second containing all the text for negative tweets. We calculate tf-idf for each document.

Test tweets are then split into their topics. We sum all the tweets for selected topic and calculate the weights of tf-idf matrix to get weights for individual class. That serve us as general public sentiment. We then calculate weights of tf-idf for each individual tweet and then multiply it with class weight.

the main differences between each methods are only in their number of classes and different evaluation methods. Subtask B uses two-point scale, containing negative and positive sentiments. For evaluation of Subtask B, we uses macro-averaged recall φ^{PN} described below. Subtask C uses five-point scale, containing also very negative, very positive and neutral sentiments. For evaluation of Subtask C, we uses macro-averaged mean absolute error (MAE^M).

6 Evaluation and discussion

We will briefly describe evaluation method for each individual subtasks that are also described in paper [3] and also our results from our implemented methods.

6.1 Subtask A: Message polarity classification

Subtask A is a single-label multi-class(SLMC) classification task. Each tweet must be classified as positive, neutral or negative. The evaluation score is mea-

sured as F_1^{PN} :

$$F_1^{PN} = \frac{F_1^P + F_1^N}{2} \quad (1)$$

F_1^P is F_1 for the positive class:

$$F_1^P = \frac{2\pi^P\varphi^P}{\pi^P + \varphi^P} \quad (2)$$

Here π^P and φ^P denote precision and recall for the positive class, respectively:

$$\pi^P = \frac{PP}{PP + PU + PN} \quad (3)$$

$$\varphi^P = \frac{PP}{PP + UP + NP} \quad (4)$$

where PP , UP , NP , PU , PN are the cells of the confusion matrix shown in table.

predicted/real	positive	neutral	negative
positive	PP	PU	PN
neutral	UP	UU	UN
negative	NP	NU	NN

First we divided our tweet messages in three documents, positive, negative and neutral document. Evaluated TF-IDF on training documents and used it on test data. We got accuracy of 38.98 % and

$$F_1^{PN} = 0.35 \quad (5)$$

Second we used random forest classifier and used our TF-IDF weights on it. The result may vary but the end accuracy is 33.95 % and

$$F_1^{PN} = 0.26 \quad (6)$$

6.2 Subtask B: Tweet classification according to a two-point scale

For subtask B each tweet must be classified as either positive or negative. For this subtask they have adopted macro-averaged recall:

$$\varphi^{PN} = \frac{1}{2}(\varphi^P + \varphi^N) = \frac{1}{2}\left(\frac{PP}{PP + NP} + \frac{NN}{NN + PN}\right) \quad (7)$$

φ^{PN} ranges from 0 to 1, where a value of 1 is the best score, while 0 means that classifier misclassified all items. This method is better than classic accuracy because its more robust to class invariance. The evaluation for subtask B is evaluated independently for each topic. The results are then averaged across topics to yield the final score.

Our results were 0.536 with φ^{PN} equation, with baseline score 0.500. We can conclude that our method is performing better than a naive classifier. If we were competing in 2016, we would get in 17th place out of 20 teams.

Our results were 0.350 with F_1^{PN} equation. We have reached the baseline score, which is 0.255, and we can conclude that our method is working. If we were competing in 2016, we would get in 32nd place out of 35 teams. We tried to improve our results with random forest classifier, but we have achieved worst results, which are 0.263, but still better than baseline score.

6.3 Subtask C: Tweet classification according to a five-point scale

Subtask is an ordinal classification task, in which each tweet must be classified into exactly one of the classes that are highly-positive, positive, neutral, negative and highly-negative. Classes are represented as numbers +2, +1, 0, -1, -2 in mentioned above order. For evaluation we use macro-averaged mean absolute error (MAE^M):

$$MAE^M(h, Te) = \frac{1}{|C|} \sum_{j=1}^{|C|} \frac{1}{|Te_j|} \sum_{x_i \in Te_j} |h(x_i) - y_i| \quad (8)$$

where y_i denotes the true label of item x_i and $h(x_i)$ is its predicted value. Te_j denotes the set of test document whose true class is c_j and $|h(x_i) - y_i|$ denotes the distance between classes $h(x_i)$ and y_i . The advantage of MAE^M over standard MAE is that it is robust to class imbalance. Unlike the previous mentioned methods, the lower the score value is, the better results are.

Our results with $MAE^M(h, Te)$ equation were 0.985 with a baseline score 1.200 where lower number is better. This was our most successful method and we would achieve 6th place out of 12 teams.

6.4 Subtask D: Tweet quantification according to a two point scale

Subtask D assumes a binary quantification setup, in which each tweet is classified as positive or negative. The task is to compute an estimate of the distribution of each of the classes. For evaluation there is used normalized cross-entropy, known as Kullback-Leibler Divergence (KLD):

$$KLD(q, p, C) = \sum_{c_j \in C} p(c_j) \log_e \frac{p(c_j)}{q(c_j)} \quad (9)$$

KLD is a measure of the error made in estimating a true distribution p over set of classes (C) by means of a predicted distribution q . KLD ranges between 0 and $+\infty$, where lower value is better. KLD is computed individually for each topic, and results are averaged to yield the final score. We didn't participate in this task.

6.5 Subtask E: Tweet quantification according to a five-point scale

Subtask E is an ordinal quantification (OQ) task, in which each tweet belongs exactly to one of the classes : highly positive, positive, neutral, negative, highly negative. The measure used for OQ is the Earth Mover’s Distance (EMD):

$$EMD(q, p) = \sum_{j=1}^{|C|-1} \left| \sum_{i=1}^j q(c_i) - \sum_{i=1}^j p(c_i) \right| \quad (10)$$

This is also a measure of error, therefore lower values are better. EMD is computed individually for each topic, and results are then averaged across all topics to yield the final score. We didn’t participate in this task.

7 Conclusion

Extracting sentiment from messages to get context of a message is one of the greatest tasks of Natural Language Processing. We have shown here that with data that we have at our disposal and tools which are evolving every day, we can easily start getting results from NLP methods for sentiment extraction. We got fairly good results which lead us to other ideas for improving the sentiment extraction. As we saw we could modify TF-IDF weights with some custom sentiment gains like emoticons and profanity words. With adding custom labels like this and improving the accuracy we could start monitoring trends on Twitter and learning to predict the atmosphere around public figures and companies on Tweeter. Further work can be made in understanding irony and jokes which were not taken in consideration.

References

1. D. Vilaresa, Y. Dovala, M. A. Alonsoa, and C. Gómez-Rodríguez, “Lys at semeval-2016 task 4: Exploiting neural activation values for twitter sentiment classification and quantification,” in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, San Diego, US, 2016.
2. G. Briones, K. Amarasinghe, and B. T. McInnes, “Vcu-tsa at semeval-2016 task 4: Sentiment analysis in twitter,” *Proceedings of SemEval*, pp. 215–219, 2016.
3. P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, and V. Stoyanov, “Semeval-2016 task 4: Sentiment analysis in twitter,” in *Proceedings of the 10th international workshop on semantic evaluation (SemEval 2016)*, San Diego, US (forthcoming), 2016.
4. SemEval-2017 Task 4: Sentiment Analysis in Twitter,
<http://alt.qcri.org/semeval2017/task4/>