# Twitter Sentiment Diffusion

Matthias Baetens (s142485) & Karol Dzitkowski (s142246)

***Abstract*—We build software to evaluate the most important features of a Tweet and their influence on the number of retweets. We used both the standard Tweet-features as well as a calculated value for the sentiment of a Tweet. We used a number of different Machine Learning models and algorithms including neural networks to compare the performance of these methods. Our system can be dynamically accesed using a webpage which is able to download new Tweets, run the different Machine Learning algorithms, perform analysis and generate relevant charts.**

## I. Introduction

One of the most important features to measure the popularity of a Tweet is the number of retweets. Next to the number of favorites, which counts how much people like a post, the number of retweets counts the number of times another user reshared the post, and thus wants to identify himself with the post and wants to share it with other. This means it is very interesting to research the possibility to optimize Tweets in order to get more retweets and to spread your message.

Tweets not only consists of the message itself: they have a huge amount of metadata:

- The time of creation of the Tweet and the user profile.
- The location.
- Whether there is a URL, an image, ...
- The hashtags (and the amount of hashtags)
- The number of followers and friends
- ...

Using the text, it is possible to calculate a certain sentiment for each Tweet; which can also be seen as a feature of the Tweet. For example: a Tweet with "awesome day" will have a more positive sentiment value than a tweet with "bad day".

We concentrated on building a basic system that downloads Tweets relevant to a certain query, calculate a sentiment and save them to a database. We implemented 6 different Machine Learning algorithms: 4 for classification and 2 for regression. The classification is used to classify tweets in a certain sentiment class using the favorite count of the Tweet, followers count of the user, retweet count of the Tweet and age of the Tweet as an input. The regression algorithms are used to predict the number of retweets based on the favorite count of the Tweet, followers count of the user, calculated word sentiment and the age of the Tweet. The results can be accessed through a website, which is implemented using Django.

## II. Related work

In Suh et. al. [1] the authors tried to quantitatively identify factors that are associated with retweeting. They split up the factors in 2 classes of features: content features and contextual features and found that for the content features URLs and hashtags seemed to have an influence on the retweet rate and for the contextual features, the number of followers and followees and the age of the account seemed to have an influence.

Dan Zarrella [2] found that users with more followers indeed get more retweets, but there are certain users without a lot of followers who get a lot of retweets, so the content of the tweets must be of some importance too. He also found that there were significantly more links in the retweets than in the tweets (56.69 % versus 18.96 %). Novelty ("newness" of the ideas and information presented) also turns out to be an important feature. The late afternoon until night (3 PM until midnight) is the most popular time to retweet.

## III. Software

In Figure 1 you can see a coarse overview of our software. In this section we will describe the different parts of our software and what their respective functionalities are.

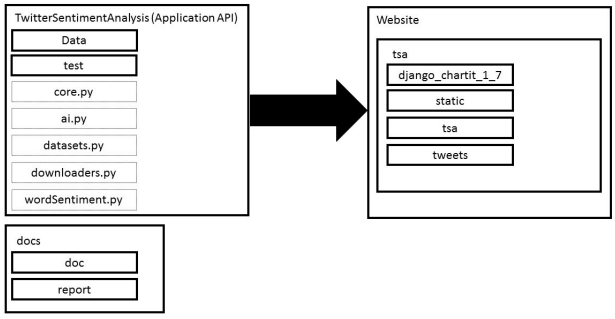


Fig. 1. Overview of the built software.

### A. The main logic/API (TwitterSentimentAnalysis)

The *TwitterSentimentAnalysis*-package contains all the logic and processing of data of our application.

*1) Data:* This folder contains the data used in the application:

- corpus.csv: the downloaded file containing records with the topics, sentiment rating and Tweet-id specified.
- words.txt: the AFINN wordlist containing different words and their sentiment value. This file is used to calculate the sentiment of our Tweets.
- ai-folder: contains the saved (and trained) Artificial Intelligences used on the website.

*2) test:* This folder contains all the tests written to test the logical part of our software.

*3) core.py:* core.py takes care of all initializing and sets up connections to Twitter and our database. It reads the necessary parameters from the *configuration.cfg*-file.

*4) ai.py:* This file contains all the Machine Learning algorithms used to predict the sentiment and the retweet count. We have implemented 6 different Machine Learning algorithms, 4 to classify the Tweets in a certain sentiment class and 2 to predict the retweet count. Classification:

- MultiClassClassificationNeuralNetwork: uses the Pybrain library [3] to construct an Artifical Neural Network with 2 hidden layers and 4 and 9 neurons in the first and second layer respectively.
- SimpleClassificationNeuralNetwork: uses the native Pybrain NNclassifier class from Pybrain tools to set up an Artificial Neural Network for classification.
- NaiveBayesClassifier: this class implements Machine Learning using the Naive Bayes Classifier from the NLTK package [4].
- MaxEntropyClassifier: this class implements Machine Learning using the Maximum Entropy classifier from the NLTK package.

Regression:

- SimpleRegressionNeuralNetwork: uses the native Pybrain NNregression class from Pybrain tools to set up an Artificial Neural Network for regression
- LinearRegression: this class implements regression using the LinearRegression class from the scikit-package [5].

All the models can be trained, evaluated (with or without using crossvalidation), saved and loaded.

*5) datasets.py:* These classes take care of the data preprocessing for the regression and classification problems. The sentiment of a Tweet gets calculated using the wordlist the following way: we search for every word in the Tweet if it is part of the wordlist and if it is, it is added to the total sum. This sum is divided by the number of words in the Tweet that are in the wordlist. This renders a result between -5 and 5. The result is then scaled to a value between 0 and 4. If there is a manual grade available (positive or negative, only for the Tweets in the downloaded file), the wordsentiment gets multiplied by 1 or -1, depending if the manual grade is positive or negative. Else, it just takes the sign it had in the beginning. This way we get 9 classes of sentiment, ranging from -4 (highly negative) to +4 (highly positive). For the classification in the different word sentiment classes the following features of the Tweets are used:

- Favorite count: the number of times a Tweet is favorited.
- Followers count: the number of followers the user who tweeted has.
- Retweet count: the number of times the Tweet has been retweeted.
- Age of the tweet: how old the Tweet is.

For the regression of the retweet count the following features of the Tweets are used:

- Favorite count: the number of times a Tweet is favorited.
- Followers count: the number of followers the user who tweeted has.
- Word sentiment: the word sentiment calculated as specified above.
- Age of the tweet: how old the Tweet is.

*6) downloaders.py:* This class takes care of downloading the Tweets using Tweepy [6]. It takes into account the rate limits of the Twitter API so our downloads do not get blocked. Tweets are saved to a specified MongoDB table using Pymongo [7], using a tag as a label when we save. This class also implements the possibility to undersample the data: because the majority of the Tweets has a sentiment of 0, the Artificial Intelligence is trained in a wrong way (a classifier which predicts all tweets to be 0 will have a very low training error and maybe even test error, although we this classifier will be useless). The undersampling will undersample data with a sentiment close to 0 (standard treshold is set to be 0.25).

*7) wordSentiment.py:* In the WordSentimentAnalyzer class the sentiment gets calculated like stated above using the AFINN lexicon [8].

### B. The website (site)

This folder contains all the code for running the site. We included the django_chartit_1_7 package, because it had to be adapted in order to work together with package we wanted to use for our charts. This package is thus not completely written by ourselves, but adapted from [9]. It contains all the logic of the different views of our site: home, analysis, statistics and contact.

## IV. Results

In this section we will talk about the results we had using our software. In this section we concentrate on results gathered from the downloaded set of Tweets, because we are more certain about the sentiment tags given manually to these Tweets, than the basic calculation of the sentiment used in our system. The error rates the different Machine Learning algorithms obtain are gathered in Table I for the classification of the sentiment and Table II for the prediction of the retweet count using regression.

| | *MultiClassClassification Neural Network* | *SimpleClassification Neural Network* |
|---|---|---|
| **Error rate** | 21.07 | 19.63 |
| | *Naive Bayes Classifier* | *Maximum Entropy Classifier* |
| **Error rate** | 25.96 | 22.97 |

TABLE I
Error rates of the Machine Learning algorithms for classification.

| | *Linear Regression* | *SimpleRegression Neural Network* |
|---|---|---|
| **Error rate** | 38.77 | 11.11 |

TABLE II
Error rates of the Machine Learning algorithms for regression.

### A. *Code checking*

### B. *Testing*

## V. Discussion

## VI. Conclusion

We delivered a basic system which is able to download Tweets, uses Machine Learning to classify the Tweets in different sentiment classes and to predict the retweet count using regression and which shows the results on a webpage. There are a lot of extensions and improvements possible in our system, but because of the flexibility of our system, this is possible. One of the most important things for future work is the improvement of the sentiment calculation: we could also use Machine Learning to calculate the sentiment value, taking into account the context wherein the words occurs; we could for example give different sentiment values to the words "bad" and "really bad". Another possibility to improve the sentiment value calculation could be to use more wordlists and compare their performance. We could also improve our results by using our API to analyze a larger database instead of just using the website for our results. Our data preprocessing scheme could also be improved to include more advanced ways of dealing with the large sentiment zero class. Some of the features we used for Machine Learning can also be transformed to improve the performance: for example, the number of followers or retweets can quickly become a large number for popular accounts (we think of millions of followers and thousands of retweets). Taking the square root or the log of these values could improve the results of our Artificial Intelligence. We think this project is a good basis for further exploration of this field and a lot of interesting results can emerge, using the right techniques.

## References


[1] B. Suh, L. Hong, P. Pirolli, and E. H. Chi, "Want to be retweeted? large scale analytics on factors impacting retweet in twitter network," pp. 177–184, Augustus 2010.

[2] D. Zarrella, "Science of retweets," 2009. [Online]. Available: http://danzarrella.com/science-of-retweets.pdf

[3] T. S. et al, "Pybrain," 2010. [Online]. Available: http://pybrain.org/

[4] E. Loper, "Natural language toolkit." [Online]. Available: http://www.nltk.org/_modules/nltk/classify/naivebayes.html

[5] started by David Cournapeaul, "Scikit-learn: Machine learning in python." [Online]. Available: http://scikit-learn.org/stable/modules/linear_model.html

[6] J. Roesslein, "Tweepy." [Online]. Available: http://www.tweepy.org/

[7] B. Hackett, "Pymongo." [Online]. Available: http://api.mongodb.org/python/current/

[8] F. A. Nielsen, "Afinn lexicon." [Online]. Available: https://gist.github.com/1035399

[9] P. Gollakota, "Django chartit." [Online]. Available: https://github.com/pgollakota/django-chartit

# Appendix A
# Code listings

## Listings

## APPENDIX B
## AUTOMATIC GENERATION OF DOCUMENTATION

Demontration using epydoc:

```
epydoc --pdf -o /home/fnielsen/tmp/epydoc/ --name RBBase wikipedia/api.py
```

This example does not use `brede_str_nmf` but another more well-documented module called `api.py` that are used to download material from Wikipedia.