Mining tweets to predict their popularity

Robin Hahling
University of Geneva,
Computer Science Department
Carouge, GE, CH
hahlinr0@etu.unige.ch

Kevin Gillieron
University of Geneva, Computer Science Department
Carouge, GE, CH
gilliek0@etu.unige.ch

ABSTRACT

In this paper we experiment several classifiers and features extracted from tweets in order to predict if a tweet will be retwitted or not. To do so, we analyzed data extracted from Twitter related to the keyword "shampoo", thus related to a community around this theme. Our approach focuses on extracting relevant features from tweets and using them to train several different classifiers such as Support Vector Machine, Decision Tree, Linear Discriminant Analysis, Maximum Entropy and Naive Bayes. We validate our approach by performing a cross-validation and comparing the classifiers in an algorithm tournament. The significance of our result is validated by the McNemar test with Bonferroni adjustment. Our result allows us to predict if a tweet will be retweeted with a maximum accuracy around 90%.

INTRODUCTION

Twitter is a microblogging system in which users cand send, reply, or forward (retweet), short messages up to 140 characters between them. Typically, users are inter-connected by messages and they form social network communities in which different topics are discussed. Messages which are retweeted several times are thus more likely to represent important topics that people like to discuss. In this project, we analyzed twitter data and built models for solving the following task: to predict if a tweet will be retweeted or not, i.e. to predict if a tweet can be considered as popular or not insight a community.

To do so, we had to develop our own data mining tool and propose an approach to solve the problem.

Our analysis was made on a recent collection of 42960 tweets, all of which are related to the global topic of *shampoo*. Due to the limitations of the new Twitter API v1.1, we were not able to fetch any more tweet than that. They cover a range of seven days (from the 13th of June to the 20th of June 2013).

Each tweet is characterized by some features which we used, along with others we created, mostly built on the tweet's content, to compute our prediction.

Our approach was to first extract informative features from the tweets and then use them into various classifiers to do our prediction. We determined which features were the most informative and which classifier performed the best by doing algorithm tournament.

TOOLS USED

`Python` was our language of choice to implement our tool to predict if a tweet will be retweeted or not. Multiple libraries were used:

- matplotlib: to plot 2D graphs of the ROC curve.
- nltk: for stemming and various language processing functions and some classifiers.
- numpy: to use special array objects.
- scipy: for some numerical routines.
- scikit-learn: for some classifiers.
- twython: to extract tweets from Twitter using their API.

Due to some libraries being not compatible with `Python 3`, we had to stick with `Python 2`.

We were able to develop a comprehensive tool that is easy to extend by taking advantage of object oriented programming. Adding new features or new classifiers can be done very easily thanks to the concept of inheritance.

The figure 1 shows the global architecture of the interesting part of the code.

Our tool is able to perform various tasks such as classification using numerous classifiers and features, algorithm tournament and cross-validation. It is also able to plot ROC curve and Decision Trees, randomizing the dataset and taking the advantage of multiprocessing for performing hard computing tasks.

In order to let the scientific community, or any people, use it, we released it under the terms of the permissive BSD license. Being written in `Python`, it is available for any operating system on which `Python` can run. We, however, only tested it on Linux.

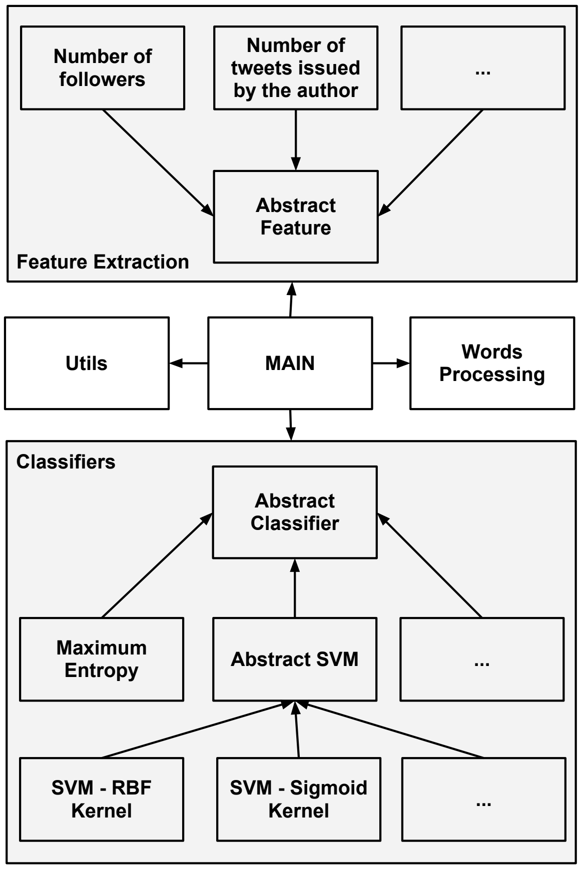

Figure 1. Source code architecture

DATASET PREPARATION

To analyze the tweets, we needed to collect a dataset of tweets. We collected tweets related to the keyword *shampoo*. Due to the limitations of the free Twitter API, we were only able to collect tweets for a recent period of time. The dataset comprises 42960 tweets from which 59.78% are retweets.

In order to extract some features, we had to do some preprocessing on the dataset. To compute the term frequency - inverse document frequency, the occurrence of words in the tweets needed to be computed.

To do so, we had to first create a corpus and then compute the occurrence of each word. We used Penn Treebane Part-of-Speech Tags in order to remove insignificant words so that only nouns, verbs and adjectives were kept from the text of the tweets. URLs, stop words or other special characters were obviously removed too. Once we had built the corpus, we created a file containing each words of the corpus and their occurrence throughout the whole dataset's tweet's text.

These words occurrences were later used in order to compute words frequencies used in TF and TF-IDF features.

The labels, i.e. a flag telling if a tweet is a retweet, extracted from the tweets have an entropy of 0.97.

FEATURES DESCRIPTION

In this section, we describe the features that we used in our experiment.

Some features are actually more significant than others, i.e. they provide a better information in order to tell if a tweet will be retweeted or not. The more features we use, the more the computation becomes complex. This increased complexity is more significant with some classifiers than others but this means nevertheless that we need to circumscribe the number of our features. Furthermore, some features may provide such an irrelevant, or even non existent, increase of useful information that they shall just be removed.

In order to perform the feature selection for classification, various methods such as information gain (IG) and chi-square (CHI) can be used. We used the latter to determine which features were relevant for our case.

The table 1 lists all the features, order from the most significant to the less significant. Features that have been dropped are in italic font. The removal criterion was the $p - value > 0.0$. When taking those features off, the accuracy increases by several points of percentage.

#	**Features analyzed**	**chi-square**	**p-value**
1	# of followers	11332870	0.0
2	# of tweets issued by the author	1248000	0.0
3	# of user's friends	998714	0.0
4	is the tweet a retweet	20615	0.0
5	# of words in the tweet	10095	0.0
6	# of users mentioned	5957	0.0
7	is the tweet a reply	3030	0.0
8	# of times "favorited"	2007	0.0
9	has an URL	790	0.0
10	# of hashtags	219	0.0
11	*is the author a verified account*	18	1.8e-05
12	*tweet's age*	16	7.7e-05
13	*highest term frequency (TF)*	NaN	NaN
14	*TF-IDF feature*	NaN	NaN

Table 1. This table lists the features used in our work.

The most significant feature is the number of people that the user follows. This value is collected from the *User* object attached to a tweet.

The number of tweets (including retweets), apart from the one analyzed, issued by the author comes at the second place. This information is provided by the attribute "statuses_count" of the *User* object.

In third place is the number of friends (i.e. the number of users following the author). This value is provided by the *User* object.

As said before, a tweet can be a retweet itself. Being at the fourth place, this feature is still sufficiently significant. This information was extracted from the text of the tweet which appears as a citation of the original tweet in the form "RT @username:".

The number of words in a tweet comes afterwards. The tweet's text was tokenized prior to count the number of words.

The feature in sixth position gives the number of users mentioned in a tweet. This information is computed based on the list of "user_mentions" given by the *User* object.

The next feature determines whether a tweet is a reply to another tweet.This information is provided by the attribute "in_reply_to_user_id" which contains a non-null value if the tweet is a reply.

The feature number 8 corresponds to the number of times a tweet has been "favorited" by other users. This value is given by the attribute "favorite_count" of a tweet object.

The following feature is based on the URLs found, or not, in a tweet The list of URLs is provided by the *Tweet entities* object.

The number of hashtags feature is simply the number of hashtags found in a *Tweet* object.

The remaining features have been disabled.

The feature number 11 indicates if the author of the tweet has a verified account. Such accounts establish the authenticity of the user's identity. This feature would probably have been much more significant if the dataset was not limited by the theme of *shampoo.*

The tweet's age, i.e. the number of days passed since the tweet was posted, is not surprisingly insignificant.

We were unfortunately not able to evaluate the significance of the term frequency (TF) and the the term frequency - inverse document frequency (TF-IDF). The chi-square obtained was NaN and therefore the $p - value$ too. Moreover, we were not able to compute the TF-IDF at all due to a bug in the *scikit* library that we use. We are pretty disappointed by this fact because the TF and especially the TF-IDF features would probably have been very good candidates.

We would have loved to try other features. Among those we wanted to implement but did not because we were short on time were the author's account age, the number of tweets per day.

CLASSIFIERS

In this work, we used and compared the 5 following classifiers :

- Naive Bayes [5], [6]
- Support Vector Machine (SVM) [2], [6]
  - with a RBF Kernel
  - with a Sigmoid Kernel
- Maximum Entropy (MaxEnt)[5]
- Decision Tree [6]
- Linear Discriminant Analysis (LDA) [6]

We excluded the Support Vector Machine from the cross-validation and the algorithm tournament because of its high computation time.

EXPERIMENT

When ran with all the options activated, our tool first collects the data from a `json` file containing our dataset, then loads the words occurrences from a file which has been previously generated and computes the term frequency as well as the term frequency invert document frequency.

Once done, the features are extracted and the instances and labels are built as lists. The entropy is computed over the labels.

From that point, classification, cross-validation and algorithm tournament are computed.

Classification

To perform the classification the train set is set as 75% of the instances, the test set being the 25% remaining. In the same way, the train labels are set as 75% of the labels and the test labels are the 25% remaining.

Then, the classification routine is run. It iterates over all the selected classifiers and computes their accuracy and predictions.

The table 2 shows the accuracy obtained during the classification process. The best ones are the *Maximum Entropy*, *LDA* and *Naive Bayes.*

Note that we ran the SVMs classifiers we only the three most significant features. This is due to the fact that giving too many features to those classifiers makes their task to find a separating hyperplane very difficult, thus requiring a huge amount of computation time for a pretty poor result.

We also implemented SVMs classifiers with linear and polynomial kernels. However, one problem with the polynomial kernel is that it can be affected by numerical instability. Thus, we did not use them.

Classifier	Accuracy
MaxEnt	91.99%
LDA	91.84%
Naive Bayes	91.77%
Decision Tree	86.97%
SVM (with RBF Kernel)	60.08%
SVM (with Sigmoid Kernel)	59.78%
Majority Vote	59.78%

Table 2. Accuracy of the classifications

Cross-validation

Once the classification is done, the cross-validation is computed. It allows us to validate our classification results.

For the task, the data is partitioned into complementary subsets. The analysis is performed on the training subset while it is validated by the testing subset. Ten rounds are performed using a partition of 10% - 90%. The results of the cross-validation are then being used to by the algorithm tournament.

The average of the results from the cross-validation for each classifier is shown in table 3.

Classifier	Average accuracy
MaxEnt	92.10%
LDA	92.04%
Naive Bayes	92.00%
Decision Tree	87.35%
Majority Vote	60.91%
SVM (with RBF Kernel)	N/A
SVM (with Sigmoid Kernel)	N/A

Table 3. Average accuracy gotten from the cross-validation

By comparing these results with the one from the classification, we can notice that they are very close to each another. The small difference is explained by the different partitioning of the dataset as the train set was less than half of the train set of the classification. Thus, the results of the cross-validation validates the results obtained in the classification step.

There is no results for the two SVMs classifiers as their computation is very long and it would need to be computed ten times. However, we can expect similar results.

Algorithm tournament

The algorithm tournament compares the results of the different classifiers, each one against each other. Due to the lack of having a powerful enough computer, we were not able to use the support vector machine classifiers in the tournament as the computation times is quite long with such a dataset and having so many features to use.

The table 4 shows the results of each round of the tournament. It gives the winner, the *Chi-Square* and the *p-value*. We defined the *Chi-Square* as follow :

$$\chi^2 = \frac{(|b-c|-1)^2}{b+c} \tag{1}$$

The McNemar test with the bonferroni adjustment says that the comparison is significant if :

$$p - value < \frac{\alpha}{n} \tag{2}$$

Where $\alpha = 0.05$ and n is the number of duel done during the tournament. In our case, $n = 10$, so the *p-value* must be smaller than 0.005. As shown in the table 4, all of our *p-values* are smaller than 0.005, therefore, these results are statistically significant.

Classifiers	Winner	Chi-Square	p-value
NB vs DT	NB	1210.79	0.0
NB vs MaxEnt	MaxEnt	16.32	5.33e-05
NB vs MV	NB	13149.10	0.0
NB vs LDA	LDA	18.05	2.15e-05
MaxEnt vs DT	MaxEnt	1330.43	0.0
MaxEnt vs MV	MaxEnt	12895.79	0.0
MaxEnt vs LDA	MaxEnt	10.06	0.0015
DT vs MV	DT	7440.81	0.0
DT vs LDA	LDA	1239.25	0.0
MV vs LDA	LDA	13174.91	0.0

Table 4. This table contains the results obtained during the algorithm tournament

The table 5 shows the ranking of the tournament. The winner of the latter is the Maximum Entropy classifier while the loser one is the Majority Vote. The Decision Tree has a quite low score, which is normal due to the dimensionality of the problem.

Classifier	Score
MaxEnt	37.5
LDA	31.0
Naive Bayes	21.5
Decision Tree	10.0
Majority Vote	0.0

Table 5. This table shows the final ranking of the tournament.

CONCLUSION

We tested and compared several classifiers with the 12 features extracted from the dataset. After the algorithm tournament and the McNemar test, we can conclude that the Maximum Entropy classifier, the Linear Discriminant Analysis (LDA) and the Naive Bayes classifier are good classifiers for the task of predicting if a tweet will be retweeted or not. Trained with our features, we obtain accuracies greater than 91%.

REFERENCES

1. Cha, M., Haddadi, H., Benevenuto, F., and Gummadi, K. P. Measuring user influence in twitter: The million follower fallacy. In *4th international aaai conference on weblogs and social media (icwsm)*, vol. 14 (2010), 8.

2. Cristianini, N., and Shawe-Taylor, J. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

3. Gama, J. *Knowledge discovery from data streams*. Citeseer, 2010.

4. Hastie, T., Tibshirani, R., Friedman, J., and Franklin, J. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer 27*, 2 (2005), 83–85.

5. Jurafsky, D., and James, H. *Speech and Language Processing An Introduction to Natural Language*

Processing, Computational Linguistics, and Speech. Pearson Education,, 2000.

6. Kalousis, A. Introduction to data mining and knowledge discovery in data. Course given at the university of Geneva, Computer Science department, 2013.

7. Mitchell, T. M. Machine learning and data mining. *Communications of the ACM 42*, 11 (1999), 30–36.

8. Platt, J., et al. Sequential minimal optimization: A fast algorithm for training support vector machines.

9. Russell, M. A. *Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites*. O'Reilly Media, 2011.

10. Spinosa, E. J., de Leon F de Carvalho, A. P., and Gama, J. Olindda: A cluster-based approach for detecting novelty and concept drift in data streams. In *Proceedings of the 2007 ACM symposium on Applied computing*, ACM (2007), 448–452.