Twitter Trending Topic Classification

MES: Maria Ludovica Costagliola - Emanuele De Santis - Serena Ferracci

Abstract— The Web Information Retrieval course allowed us to develope the described project. It is about classification of topics that are trending of twitter in a given moment. We have dealt with (i) text based classification using single tweets (ii) network based classification exploiting the graph structure of Twitter.

The used dataset is composed by tweets divided by trending topics. We classify the trending topics into 8 categories: *Event, Health, Movie, Music, Politics, Science, Society* and *Sport*.

The first approach gives good results in terms of precision and recall. Instead, the second approach does not give conclusive results because it requires too many resources to be implemented on a single machine.

1 Introduction

We chose this project because Twitter is one of the most popular social network. It is used every day by millions of users expressing their opinions about several fields. When an user looks for something, the first thing Twitter displays to him is the list of trending topics of the moment. Often the user can not know what the topic is about, so it has to manually search for tweets belonging to that trending topic to better understand it.

It is interesting for the user to have a way to know what the Trending Topic is about without further searches. Our work tries to replicate the results obtained by [3]. The general categories used in this project are 8, named: *Event, Health, Movie, Music, Politics, Science, Society* and *Sport*.

2 Related Work

The paper we refer to is the work done by Lee et al. for tweets classification [3]. They have used 18 general categories to classify each trending topic. They address the problem following two different approaches. The first one is a supervised learning technique, named Multinomial Naive Bayes. Instead, the second one is network based and it uses Personalized PageRank to compute the top-k influencers and then it computes the intersection between the influencers to classify the new trending topic.

All trending topics they used were downloaded from *what the trend*.

3 Dataset

Since it was not possible to get a suitable dataset for our purpose, we needed to build it manually using Twitter APIs [1] through Tweepy [2].

In `tweets_retrieve.py` we first retrieved trending topics from USA using USA WOEID (Where On Earth IDentifier, that is 23424977 for USA). Then, for each trending topic, we queried Twitter to get all tweets belonging to the given topic. We pay attention to retrieve the entire text of the tweet; in case of retweeted statuses we consider only the retweeted text. Twitter imposes some limitation on GET requests, in particular we could retrieve 180 tweets every 15 minutes. So, we have to handle exceptions fired due to the reached limit. In order to submit a request we have to set up Tweepy authorization handler with our consumer key and our access token we take from our twitter developer account. As result, we get a file for each tweet where its name is the ID of the author and the body is the text of his tweet. This file is placed in a folder whose name is composed by the name of the trending topic. At the end, we collect about 20.000 tweets belonging to 139 trending topics.

After this first phase, we had to retrieve the link structure for the users that wrote the tweets collected in the previous phase. In order to do so, we developed two python files, named `followers_retrieve.py` and `friends_retrieve.py`. We used Tweepy to retrieve all followers and friends for each author ID being aware of all exceptions. Since the limitation, in this case, is 15 queries every 15 minutes, we had to find a way to speed up the process. To do so we decided to parallelize the computation using more than one key (precisely 6 keys), contrarily with respect to the previous retrieving phase, and working in data separation. We associated each key to a subset of IDs to be processed, this mapping was done using a simple hash function. We saved the list of followers and the list of friends for an user *x* in the files named `x`

`- followers.txt` and `x - friends.txt`. These two files are saved in the same folder that contains the tweet tweeted by x.

We faced three main problems to retrieve the dataset:

- some users changed their privacy settings between the first and the second phase of our retrieving. For this reason we couldn't get neither their followers nor their friends.

- a subset of users have a plethora of followers, of the order of millions of users. It took a lot of time to complete the retrieving of their followers, sometimes almost an entire day at full capacity. It also happened that the execution stopped due to connection problems or other unpredictable events, so it was not possible to retrieve the entire list of IDs.

- there wasn't a retrievable ground-truth for the trending topics we collected.

The last problem was the only one that we could manage to solve.

We tried to automatically get a classification using some pretrained machine learning algorithms, but when we manually checked the classification done, we saw that almost all the predictions were wrong.

The solution that we came up with was to label every trending topic by hand. We took one directory at a time, we read some tweets to understand what the topic was about, we assigned to the directory a single category among the 8 defined.

4 Classification using Naive Bayes

Multinomial Naive Bayes is a very useful Machine Learning algorithm when we are dealing with text classification. Naive Bayes is based on the assumption that the probability that a document (composed of terms $(t_1, ..., t_T)$) belongs to a class c_k , namely $P(c_k|t_1, ..., t_T)$, can be written as $P(c_k) \cdot \prod_{i=1}^{T} P(t_i)$. The probability that a term is in a class is conditionally independent from the probability that another term is in the same class.

So Naive Bayes classifier makes a prediction $\hat{y}$ using the following formula

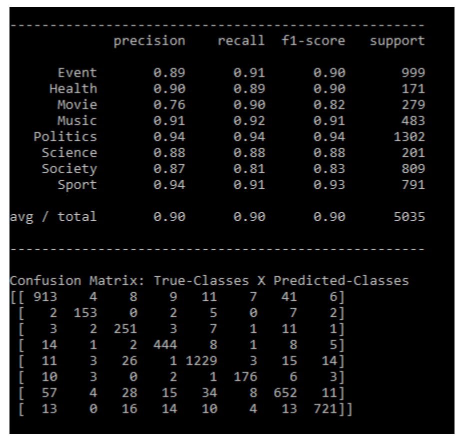
$$\hat{y} = \underset{k \in \{1,...,K\}}{\arg\max}\, P(c_k) \cdot \prod_{i=1}^{T} P(t_i|c_k)$$

We rearranged the data in the dataset in order to have a folder that contains 8 subfolders (one for each category we considered). Each one contains all the tweets belonging to a trending topic that belongs to the category of the subfolder. This operation is done by `training_set.py` python script.

We developed also `naive_bayes_no_stemmming.py`. It takes as dataset the folder yet mentioned.

`sklearn` allows to make an automatic division of the dataset, splitting it into train set and test set (25% for the test set). The training set was converted first computing the corresponding tf-idf matrix through the `TfidfVectorizer` function. This is then used to actually transform the original dataset into the term-document matrix. The term-document matrix is used by the Multinomial Naive Bayes classifier to execute the learning process.

After the learning phase, we needed to evaluate the performances of the classifier: we applied the same transformation as before to the test set and we made the algorithm predict these instances. We get at the end the following results:



```
---------------------------------------------------------
             precision    recall  f1-score   support

      Event       0.89      0.91      0.90       999
     Health       0.90      0.89      0.90       171
      Movie       0.76      0.90      0.82       279
      Music       0.91      0.92      0.91       483
   Politics       0.94      0.94      0.94      1302
    Science       0.88      0.88      0.88       201
    Society       0.87      0.81      0.83       809
      Sport       0.94      0.91      0.93       791

avg / total       0.90      0.90      0.90      5035

---------------------------------------------------------

Confusion Matrix: True-Classes X Predicted-Classes
[[ 913    4    8    9   11    7   41    6]
 [   2  153    0    2    5    0    7    2]
 [   3    2  251    3    7    1   11    1]
 [  14    1    2  444    8    1    8    5]
 [  11    3   26    1 1229    3   15   14]
 [  10    3    0    2    1  176    6    3]
 [  57    4   28   15   34    8  652   11]
 [  13    0   16   14   10    4   13  721]]
```

Fig. 1. Classification evaluation without applying stemming

We tried, also, another version of the program, `naive_bayes_stemming.py`, where we used the stemming technique: we tokenized each word, applying an `EnglishStemmer` from `nltk` [?]. Making the vectorization phase together with stemming, we get:

5 Network Based Classification

To compute the network based classification, we first had to construct the direct graph, that represents a small subset of the Twitter's network. In the direct graph every node represents an user and every edge represents a relationship between two users. If a node has an incoming edge, meaning it is followed by another node, we call the last one follower. Instead, if a node has an outgoing edge, meaning that it follows another node, we call the last one friend. To construct the described graph, we developed the python file `pagerank.py`, in which we

```
--------------------------------------------------
             precision    recall  f1-score   support

      Event       0.77      0.89      0.83       995
     Health       1.00      0.04      0.08       174
      Movie       1.00      0.32      0.49       302
      Music       0.97      0.69      0.81       482
   Politics       0.63      0.99      0.77      1337
    Science       1.00      0.31      0.47       189
    Society       0.93      0.62      0.74       819
      Sport       0.87      0.86      0.87       737

avg / total       0.82      0.76      0.74      5035

--------------------------------------------------

Confusion Matrix: True-Classes X Predicted-Classes
[[ 888    0    0    2   86    0    8   11]
 [  44    7    0    0  113    0    0   10]
 [  37    0   98    0  136    0    6   25]
 [  36    0    0  331   85    0   16   14]
 [   9    0    0    0 1318    0    0   10]
 [  38    0    0    2   77   58    4   10]
 [  85    0    0    1  214    0  506   13]
 [  17    0    0    4   78    0    4  634]]
```

Fig. 2. Classification evaluation applying stemming to the training set

used the predefined library `NetworkX`. The library offers to the programmer several functions that facilitate the construction of the graph. Once the graph is finalized, we compute the PageRank using a function present in the mentioned library. This function takes as parameter only the graph defined as describe above and returns a dictionary of nodes with the PageRank as value. The purpose of this computation is to find the top-k influencers for each category. The top influencers are the ones that have the higher PageRank value.

REFERENCES

[1] Standard search api - twitter developers. https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.

[2] Tweepy documentation. http://docs.tweepy.org/en/v3.5.0/.

[3] K. Lee, D. Palsetia, R. Narayanan, M. M. A. Patwary, A. Agrawal, and A. Choudhary. Twitter trending topic classification. *2011 IEEE 11th International Conference on Data Mining Workshops*, 2011.