Sentiment Analysis of the Top Trends of the week in Twitter around Chicago

Done by Shivakumar Vinayagam (A23041139)

Introduction:

* Analyze a week of tweets around Chicago from twitter and get the top 5 most trending topics based on the hash tag analysis of the tweets.
* Sentiment analysis can be done using a logistic regression classifier. We train the classifier on labeled tweets and use the classifier to predict unlabeled tweets. This will help us assign a sentiment on unlabeled tweets.
* Also train a classifier on the week of tweets to help predict future trends to place a sentiment on them. This was we can improve analysis on future trends.
* Determine the Best settings for the Classifier so that it is not too biased to the data and has some decent accuracy of prediction.
* The hypothesis which I am trying is that I can use tweets without a trend in it to predict the sentiment towards the trend in new or future tweet which have it. This helps can predict on new trends and hash tags when they appear new in twitter without any prior information.
* This prediction is possible cause of the other data in the tweets containing trend which appears in the trained data.

Data:

* The data will be collected from twitter using Twitter Search API which is a REST API.
* Used the Location tag for Chicago to search the tweet and the tweet id and time of it to collect distinct tweets of one week.
* Here the Data of one week was collected and stored in file in order to reuse. There are a total of 215 files, each containing 100 tweets. The Rest API was used to retrieve the tweet by manipulating the Tweet id and date to get a distinct set of tweets .
* About 22k tweets were collected from around Chicago, then curated and stored in the files.
* Each record is for one tweet and contains a list of tweet (text), hash tags, mentions and urls each.

Example tweet :

{'text': u'RT @am_in_america: #Chicago. Love this city
\u2764\ufe0f\U0001f389\U0001f60d\U0001f64c\U0001f3fd\U0001f377\U0001f377\U0001f37
7\U0001f377. View taken from a lovely little #yogi rooftop . Missing my #ytt\u2026
https://t.co/fnfb4\u2026',
'hashtags': [u'Chicago', u'yogi', u'ytt'],
'id': 671200182604718080L,
'urls': [u'https://t.co/fnfb4sdsnG'],
'mentions': [u'am_in_america']}

* The assign_labels_store(tweets,count) function is for the purpose of helping to label tweets manually and store them under "label+count" file name.
* We manually labeled the first 500 tweets which are the most recent ones by time when collected from twitter and stored the labels in “label 500” file.
* The labels we used are of 3 classes:
    1. ‘1’ - positive
    2. ‘0’ - Neutral
    3. ‘-1’ - Negative

Methods:

* First collect a week of twitter data using the Twitter REST API, curate it into tokens, hash tags and other details for each tweet and store it in file format.
* The assign_labels_store(tweets,count) function is for the purpose of helping to label tweets manually and store them under "label+count" file name.
* Used 500 tweets to find the optimal settings for the vectorizer and Logistic Regression Classifier.
* Use SentiWordNet to classify the tweet as either positive (+1), negative (-1) or neutral (0) , since it gives more details than AFINN.
* We used both the manually assigned labels and the sentiwordnet labels to determine the settings and to check their performance.
* Then use the hash tags to get the top 5 most trending topics in the dataset collected.
* Vectorize the dataset and train the classifier without the tweets from the top trends and based on this predict the sentiment towards the top trends.

Experiments:

* The twitter search api sent only the recent tweets so used tweet id and date to get distinct tweets of one week time.

* The various setting of the vectorizer was experimented over to find the optimal values for n_fold, min_dt, max_dt, tokenize function and binary.

* The optimal setting found for the vectorizer are 5 folds, min_dt=2, max_dt=0.5, tokenize func=tokenize (w/0 punc) and binary as true.

* We implemented 5 tokenize function of varied method calling them as,

    1. tokenize(tweet['text'])
    2. tokenize_with_punct(tweet['text'])
    3. tokenize_with_not(tweet['text'])
    4. nltk_tokenize(tweet['text'])
    5. preprocess(tweet['text'])

* Preprocess also does pretty well but when training on large amount of data, there was a drop in accuracy. I haven't included the data. But that is why i decided to use tokenize function itself.

* My manually labelled data gave a poor average accuracy of 32%. This is due to the nature of the tweets and the inability of the classfier to understand sarcasm and pseudonyms of the English language which is difficult to quantify. Also this brings in the concept of human error also, as I am the one labelling the tweets and assigning values and the labels will be morally biased towards my view of tweets (Example a tweet which i will give a negative label, can be given a neutral or even positive label by another person). In order to get a more globalised view we would need to build a much much more complex Classifier or machine learning algorithm. This leads us to use some other way of labelling the tokens such as the SentiWordNet method.

* On the other hand the sentiword which gives labels based on preassigned positive and negative token values does better with an average accuracy of 72%. This gives us a better way to labelling the tweets. It also has the advantage of a globally symantic view due to its predefined nature. The clash of the different human views is also avoided. Also I can use the function to label and train on more data rather than just 500 tweets and this will increase the accuracy even more.

* As discussed above, we will be using the sentiwordnet to classify the 22k tweets.

* Also trained and predicted on entire dataset itself to get to estimate the full accuracy on already seen data. We found a vectorization of 21422 tweets with 11441 terms and an accuracy of 0.959667562943.

* Tried using tokens to find the top trends in dataset but found too much of noise to get proper results. Hence used Hashtags to get the top trends.

* Also used the top trends and exclude them from the dataset to train on remaining and predict on the excluded trends.

* Result when using tokens of tweets to find top trends where

    1. The Term chicago has count 10220.0
    2. The Term il has count 5147.0
    3. The Term in has count 4990.0
    4. The Term the has count 4979.0
    5. The Term job has count 3846.0

When i used tokens from the tweet to determing the top trends in the 22k tweets it is giving out a lot of noise. And segregating through those will take manual work. Hence decide to not use the tokens to get top trends.

* Thus used Hashtags to get the top trends and found

[(u'Chicago', 4093), (u'Hiring', 2558), (u'job', 1922), (u'Job', 1900), (u'CareerArc', 1747), (u'Jobs', 1696), (u'chicago', 777), (u'hiring', 773), (u'ncaa', 531), (u'Hospitality', 512)] as the top 10 trends.

* Here the data is much more clear and no noise. But still there are some like "job" and "Job". We removed those by using lower func, but it still doesn't eliminate tags like "jobs".

* Also using the top trends, we excluded them from the dataset to train on remaining and predict on the excluded top trends. Got an average accuracy of 0.815919242915637 for the top 5 trends and accuracy of 0.82190589520515489 for top 20 trends.

    1. top element chicago with count 5028 has accuracy=0.7365
    2. top element job with count 3822 has accuracy=0.6688
    3. top element hiring with count 3330 has accuracy=0.8423
    4. top element careerarc with count 1747 has accuracy=0.9376
    5. top element jobs with count 1696 has accuracy=0.8267
    6. top element ncaa with count 531 has accuracy=0.9699
    7. top element hospitality with count 512 has accuracy=0.2305
    8. top element ncaaw with count 472 has accuracy=0.9470
    9. top element veterans with count 431 has accuracy=0.8747
    10. top element it with count 344 has accuracy=0.8721
    11. top element sales with count 339 has accuracy=0.9292
    12. top element accounting with count 267 has accuracy=0.9476
    13. top element businessmgmt with count 247 has accuracy=0.9352
    14. top element retail with count 234 has accuracy=0.9231
    15. top element trndnl with count 221 has accuracy=0.8597
    16. top element thanksgiving with count 205 has accuracy=0.8537
    17. top element healthcare with count 171 has accuracy=0.6550
    18. top element laquanmcdonald with count 164 has accuracy=0.8354

19. top element finance with count 161 has accuracy=0.9130
20. top element nursing with count 147 has accuracy=0.6803

* This helps can predict on new trends and hash tags when they appear new in twitter without any prior information. This prediction is possible cause of the other data in the tweets containing trend which appears in the trained data.

Conclusions and Future Work:

* Used REST API to get tweet around Chicago of 1 week period.
* The optimal setting found for the vectorizer are 5 folds, min_dt=2, max_dt=0.5, tokenize func=tokenize (w/0 punc) and binary as true.
* Used the optimal setting on vectorizer and got an accuracy of 0.74049999999999994.
* Used Hashtags instead of tokens to get top trends and got "Chicago" as top. This is expected as search tags using location around a city includes geotagged tweets and the city tagged tweets mainly.
* Didn't want to bias the classifier too much towards the dataset and hence calculated the accuracy excluding the top trends to predict the top trends and got a value of 0.815919242915637.
* Also using the top trends, we excluded them from the dataset to train on remaining and predict on the excluded top trends. Got an average accuracy of 0.815919242915637 for the top 5 trends and accuracy of 0.82190589520515489 for top 20 trends.
* For future work, the ideas implemented in paper (1) can be experimented with, where we built a sentiment classifier by ourselves by using NLP concepts. Also we need more manually labeled data in order to build an efficient classifier
* The diagrams and graphs to explain my results are in the ipython notebook file. Please kindly take a look at them.
* We read a few paper and took a few ideas from them and incorporated them to our idea.

References:

1. http://www.cs.columbia.edu/~julia/papers/Agarwaletal11.pdf
2. http://marcobonzanini.com/2015/03/09/mining-twitter-data-with-python-part-2/
3. http://emnlp2014.org/workshops/CodeSwitch/scripts/TweetTokenizer.pm
4. http://nlp.stanford.edu/software/tokenizer.shtml