Wright State University

CS 7900

Final Report

A Sentiment Analysis Application for Tweets

Author:
Daniel McIntyre U00653482
Nathan Rude U00637118

Supervisor:
Dr. Tanvi Banerjee

April 16, 2016

1 Goal

Social media is a very ingrained component of our culture. In this day and age, there are social media users all over the world. In fact, social data is a very rich source of data in todays information age. For this project, we aim to leverage the power of social data from Twitter. As a micro-blogging outlet, users regularly churn out bite-sized chunks of information that can be analysed for information on just about any topic. By aggregating tweets about a certain topic from Twitter, one can get a general sentiment from the populous about that certain topic.

Since tweets can be retrieved in real time, the sentiment of current and trending topics, such as presidential debates and debate topics, can be evaluated to determine how the populous feels about each topic. Understanding and incorporating this user feedback, specifically by examining tweets with a negative sentiment, one can receive the feedback required to move towards a more popular opinion.

2 Objective

We created a web application that allows users to view the sentiment of topics on Twitter. Through the use of two APIs, we 1) retrieved tweets from Twitter and 2) performed sentiment analysis on these tweets and also aggregated the general sentiment about a topic.

3 Approach

This application can be split into two parts: a frontend for the user to select topics and view the sentiment, and a backend for the text retrieval and sentiment analysis to take place.

3.1 Frontend

The user interface was developed with HTML, CSS and Javascript, technologies that have been learned in class. Specifically we used nodejs as a server backend to host the site. We used the Express framework for routing and logic as well as Swig for the user interface templating language. Swig provided a flexible design that incorporated template inheritance and ease of use. In addition to Swig, Bootstrap was used for the CSS styling.

The user interface is comprised of two parts, the user input section, and the the results display. The user input section consists of a start date, end date, topic search, and a selection to search by topic or by tweet id. The start and end dates define the time range that returned results will occur in. The topic search input can be a tweet topic, or can be a tweet id. If the tweet id check mark is selected, a specific tweet with the specified id will be searched and if found, it's sentiment will be analysed. The tweet results section shows the tweet, the tweet sentiment and the person who tweeted it. The accumulated tweet sentiment is also displayed to the user.

Using these technologies we can create a dynamic layout that, given a search topic, can display several retrieved tweets with their individual sentiment, as well as the overall sentiment of a topic.

3.2 Backend

The backend was split into two parts: tweet retrieval and text sentiment.

3.2.1 Tweet Retrieval

The tweet retrieval was performed with the node module *twit*. This module allowed us to search twitter with query strings that encode the topic, start and end date. Using this API we can also search twitter with a specific tweet id to get the associated tweet. The twitter API does have its drawbacks however. The first drawback being that there is a usage rate limit for the search API of 180 calls per 15 minutes.

The second limitation is the page size when using the search API. The page size denotes the number of tweets retrieved per api call up to a maximum of 100 tweets. This allows us to retrieve 100 tweets with each 180 allowed API calls for a total of 18,000 tweets each 15 minutes. This might seem like a large number, however this number can be easily exhausted when querying a popular topic. For example, querying "Trump" will return 100 results within a time window of a few seconds. Querying another, less popular topic such as "Grand Canyon" will return 100 tweets within a time window of several hours (note that this is situational and the frequency of requests could change).

The third limitation is that the search API only indexes tweets within a 7 day window of when they were created. After roughly 7 days, the tweets are removed from the search index and are unable to be retrieved. This makes it impossible to retrieve tweets by topic that are further in the past.

To accommodate all of these limitations, our program only returns 100 tweets per search. However calculating the sentiment of only 100 tweets is not indicative of the overall sentiment of a topic. For popular topics that are constantly being discussed, this 100 tweet limit is a mere snapshot of a few seconds of conversation. The only way to overcome these limitations would be to use the Twitter streaming API to retrieve tweets as they occur, however there are no current plans to integrate this into our application.

3.2.2 Text Sentiment

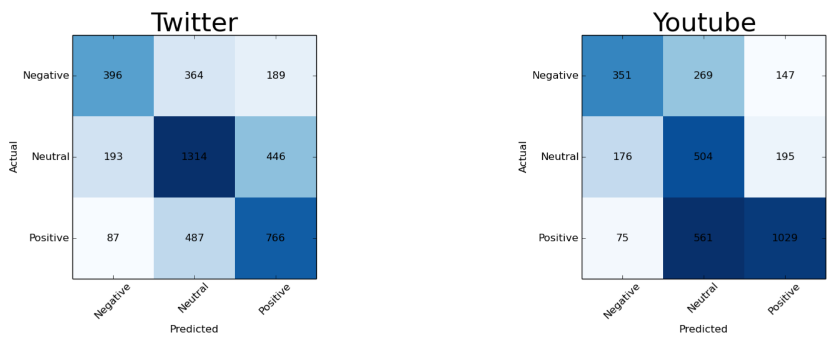
The sentiment analysis was performed with the node module *retext-sentiment* which is a plugin for the retext module. The retext-sentiment module operates at the sentence level through the word level. Each word is assigned a polarity and negation is also supported. The word "bad" would have a negative connotation, but the phrase "not bad" would not have a negative connotation. There is also built in support for emoji as well. A smiley will have a positive connotation while a frown would have a negative connotation.

The overall sentiment of a tweet would be *sentiment = good words - bad words*. This gives us a sentiment polarity that represents how positive or negative the sentiment is and, consequently, a valence of negative, neutral or positive.

One limitation of this module is that it uses a mapping of sentiment to words, so if a word is misspelled, the sentiment cannot be computed. Since this module is used on unprocessed twitter data, there is the possibility that tweets exhibiting a negative sentiment, but are not spelled correctly, will not be labelled accurately.

4 Evaluation

To evaluate the sentiment analysis component of the application we used the following two sub-datasets from the Sentiment Strength Twitter Dataset (SS-Tweet) [4]: 1) Youtube comments and 2) twitter tweets (see Table 1). Using data from two domains allowed us to better evaluate the robustness of our sentiment analysis tool.



(a) (b)

Figure 1: Sentiment Confusion Matrices

To keep the evaluation simple, we performed binary evaluation for this multiclass problem. The testing datasets were labeled with a mean positive and mean negative value for each sample. To calculate the sentiment, we define *sentiment = good - bad* and then threshold it into a class *negative, neutral, positive* based on it's value. To compute precision and recall for a multiclass problem, we transformed the problem into several one-vs-many problems, calculated the precision and recall for those, and then took the average values for the multiclass problem.

Dataset	**Samples**	**Positive**	**Neutral**	**Negative**
Twitter	4,242	1,340	1,953	949
Youtube	3,307	1,665	875	767

Table 1: Dataset class breakdown

The results of the sentiment analysis evaluation can be seen in Table 2 and Figure 1. We can see that the Twitter and Youtube datasets have similar precision and recall values, and this implies that this sentiment module is robust across different text domains. Now while the precision and recall values do not seem that high, it's important to realize that not even humans can accurately determine the sentiment of a text 100% of the time. In fact humans can only determine the correct sentiment 70-80% [1][2][3] of the time! This is because humans do not universally agree with each other on any subjective matter.

Examining the confusion matricies in Figure 1 we can see that the algorithm mostly has trouble with the neutral class. With sentiment being such a subjective issue, it intuitively makes sense that there would be more trouble evaluating whether a text is *positive vs neutral* or *negative vs neutral*. The lines can be more gray here rather than the stark contrast that a *positive vs negative* tweet would have.

Dataset	**Precision**	**Recall**
Twitter	.55	.58
Youtube	.53	.55

Table 2: Sentiment Metrics

Since the algorithms performs much better than just guessing the class (which would be a 33.3% precision), we conclude that the performance of this algorithm is adequate for the task at hand.

5 Results

put sample queries here and evaluate and analyze the results

1) gender neutral bathrooms

6 Conclusion

References

[1] brnrd.me. On social sentiment and sentiment analysis. `http://brnrd.me/social-sentiment-sentiment-analysis`. Accessed: 2016-04-16.

[2] brnrd.me. Sentiment analysis: Why its never 100 `http://brnrd.me/sentiment-analysis-never-accurate`. Accessed: 2016-04-16.

[3] Mashable. How companies can use sentiment analysis to improve their business. `http://mashable.com/2010/04/19/sentiment-analysis`. Accessed: 2016-04-16.

[4] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold. 2013.