Sam Clark
10/10/10

CSE 454 Project Proposal

Group Name(s)

EmotoTweets, EmotoTrends, EmoticonTweetTrends

Group Members

Sam Clark, Jesse Shepherd, Jon McCord, Curtis Yamanaka, Golf Sinteppadon, Michael Amorozo

Goal

Track all named entities that are co-occuring with emoticons that express happiness/sadness (ie. :), :-), :(, etc.) in Twitter status updates. Use this data to create a site that displays the most commonly co-occuring NEs in interesting ways. One example is a page displaying the top trending co-occuring NEs, where 'trending' is really some metric calculated by judging the increase in mentions of the NE. This could possibly serve as news to some (maybe the same demographic that likes to use emoticons...), but could also just be an entertaining pulse on Twitter for others.

Project in terms of smaller chunks

Adapting NLP Tools to Twitter 80 hours-

We plan on using domain adaptation techniques to create taggers that utilize both in-domain and out-of-domain data. For example: http://acl.ldc.upenn.edu/P/P07/P07-1033.pdf. This will involve manually tagging a number of tweets.

Another problem with Twitter that may need to be handled is improper capitalization. Here is one possible way for re-casing words: http://acl.ldc.upenn.edu/acl2004/emnlp/pdf/Chelba.pdf. Another option might be to just identifying improperly cased tweets and ignore them.

Also, Twitter is a international blogging service and our NLP tools will only work on English. To handle this we need to attempt to classify non-English tweets so we can filter them out. This can be done using a simple uni-gram or bi-gram language model. The training data for this can be attained from tweets written by people in cities that are known to have one predominant language (NYC, Berlin, Paris, Tokyo, etc.)

If time:
-Find common phrases that are co-occurring with NEs
    -Could be verbs, nouns, etc.
    -Might just use uni-grams and bi-grams
Example:
Paul the Octopus -> 'has died', 'died today'

Infrastructure N hours-

We would like to create a robust system for collecting, processing and storing tweets. Currently using Twitters streaming API we can collect 100-200K tweets per hour that contain emoticons.

The system needs to accomplish the following:
-Maintain a continuous http connection with the Twitter API. This will go down at least once every 48 hours and will have to be automatically monitored/reset.
-Cache recent tweets and check each new tweet to see if it is redundant (can be found in the cache). The cache will need to be maintained by removing aging tweets.
-Process tweets using libraries provided by the NLP group. Since this could potentially be a bottle neck the infrastructure will have to continually check how many backlogged tweets exist and drop some if necessary.
-Continually update the metrics for given NEs (how much they are trending) as new tweets with that given NE are added.
-Remove data from the main Tweet table as it ages.

If mapping feature is implemented:
-Maintain a table with location information for users that tweeted about each NE.

Frontend N hours-

We would like to create a very clean, fast, and interactive frontend. It will be generating its data from the continuing
-Each NE in the trends list can be clicked on resulting in the exposure of recent tweets about the NE. We plan on using AJAX in order to continually update this stream (not necessarily with new tweets, just more tweets about the NE)
-We might also graph the popularity of a NE over time to display how long it has been trending and when it started.
-One challenge is that it will have to function on multiple browsers and could potentially generate different content based on location (if we implement some sort of local trends page).

If mapping features are implemented:
-Could map where people are tweeting about a given NE from (Giants, gulf oil spill, etc.)
-Could make it possible for people to zoom in on a map and have a list of trends for the region inside the map displayed.

Project milestones

October 29th - Have basic infrastructure in place for tracking hashtags (a simpler thing to track than NEs, but would use the same infrastructure). Also have a basic FE in place to display them.
November 17th - Have all core features in place.
December 3rd - All extra features that were being considered implemented.

Test and training data

Using the Twitter streaming API as mentioned earlier.

How success will be evaluated

NLP tests

-Test accuracy of POS tagger and precision/recall for NP chunker and NER on a given set of tweets.
-Test accuracy of non-English tweet filter on a given set of tweets.

Infrastructure test

-Do simulated stress tests to see how well it could handle extreme twitter traffic (Michael Jackson death).
-Profile to find bottle necks that might need to be handled if we get a larger firehose of tweets.

Frontend test

-Check latency for serving pages under different traffic conditions.
-Check how well it is supported by different browsers.
-Test a variety of different use cases for each feature (will be set up formalized once these features have been decided on).