Siena's Twitter Information Retrieval System: The 2014 Microblog Track

Timothy LaRock, Lauren Mathews, Matthew Roberts
Dr. Darren Lim & Dr. Sharon Small
Siena College
Loudonville, NY 12211
tlarock@albany.edu, {li08math, mw20robe, dlim, ssmall}@siena.edu

ABSTRACT

As the internet dramatically changes each year, microblogs – such as Facebook and Twitter – are being used more often as a source of information exchange. Twitter users are learning about current events earlier compared to reading about it on their news feeds, as companies and celebrities continue to utilize Twitter to spread information. Information Retrieval, a topic which NIST[1] (National Institute of Standards and Technology) holds a conference for every year, involves utilizing such online environments, like microblogs, to grab as much information from these sources to find if the information can be put towards a purpose. The Microblog Track was originally introduced to TREC[2] (Text REtrieval Conference) in 2011, and selected Twitter[3] as its microblog resource. Twitter allows its users to share short, 140 character length posts with their followers, and is often used to share anything from fashion trends to the latest terrorist attacks. Due to the short length of tweets, users often utilize other ways to share more information, such as including links or images with their tweets, which has an effect on the tweet containing relevant information. Participating groups for the track were given access to a Twitter API, provided by TREC, containing a corpus of 243 million tweets scrapped from February 1st to March 31st, 2013. Each group was given a set of test topics in which to test their system, which return results for the Adhoc and/or Tweet Timeline Generation Task (TTG). In this paper, we describe five Query Expansion modules and three Relevance modules designed for the microblog track, built within STIRS. Our precision results for our adhoc run shows STIRS' average to be at 61.91% precision, with our average TTG at 85.38% precision.

1. Information on TREC's Microblog Task

Continuing onto its fourth year, TREC's Microblog Track has become one of the most popular tracks for information retrieval. Twitter, a well-known social networking site, allowed TREC to obtain an API Tweet Corpus of around 243 million tweets, obtained between February 1st and March 31st, 2013. The tweet information has usually only included user ids, dates, query times and the actual tweet content, but for 2014 was extended to include: the followers count, the statuses count, the language the tweet was in, how many times that tweet was retweeted and other ids based on these attributes. Given a set of test queries, the query would be sent through a group's system, and would output relevant information, in the form of tweets, on the topic, depending on the task requirements.

1.1 Temporally-Anchored Ad Hoc Search Task (Adhoc)

Participants were given a set of topics, gathered from the tweet collection, and asked to return the top 1000 relevant tweets for each topic. One of the most important concepts was that information returned by each system could not be older than the moment of the query time; any older tweets would be automatically judged as irrelevant. Each group was asked to return up to four runs, with each run having six fields: the topic number, an unused column, the tweed id, the rank of the tweet as determined by the run, the score given by the system, and the run tag. While both external[4] and manual runs were encouraged in past years, 2014

[1] http://www.nist.gov/
[2] http://trec.nist.gov/
[3] https://twitter.com/

[4] Evidence obtained from sources other than the official API.

is the first year that TREC has banned future evidence[5] from any runs.

1.2 Tweet Timeline Generation Task (TTG)

The TTG task was new to participants of this year's track, and was not required for all groups to attempt. In addition to retrieving relevant tweets for each topic, redundant tweets were to be removed and each run was only to return the tweets needed to correctly summarize a topic (semantic clusters[6]). This meant that each tweet must not only be relevant for the topic, but contain information that other tweets, also returned for the topic, did not have. Due to these conditions, the two main tasks within TTG were eliminating redundancy and deciding the size of the set for each topic. This presented a unique challenge because it required the system to make decisions about whether a tweet was more or less representative of the query than another tweet and to differentiate between clusters, or types, of tweet. All participating groups were asked to return runs in the same format as the Adhoc task, although the rank and score are considered negligible.

2. The System

Our team utilized an 8-processor, Dell Precision Workstation 490 for accessing the API, developing STIRS and executing various experiments for each task. Each processor was an Intel Xeon 3.00 GHz CPU, each having two CPU cores. The server had 16 GB of memory, 750 GB of hard drive space and was running KDE Plasma workspace and Eclipse Java IDE.

2.1 API

For this year's track, TREC employed the Evaluation as a Service model to give participants access to the corpus through an API, which was accessible to every group via a group ID and token password. The API was able to take a query, a time and a number of results (maximum was 10,000) and return a set of tweets indexed by Lucene[7]. Lucene naturally indexes all tweets from the entire tweet corpus and gives the tweet a score based on the words, implementing both text normalization and stemming in the process. Each returned tweet contained the topic it was considered relevant for, the tweet id, the rank and score given by Lucene, the tweet and other attributes found by analyzes the user's account or the tweet itself. While acquiring each topic's tweets, our system discarded tweets that were non-English and/or retweets (as both such types of tweets are judged automatically irrelevant). Our system wrote the tweets returned by the API into an index directory, which our system was able to easily access during runs. This index was treated as a local copy of the corpus of tweets relevant to the given queries.

2.2 Query Splitting

In past years, participating groups in the Microblog track were given the choice of downloading the full tweet corpus for their machines, which allowed them access to every tweet residing in the corpus. Since this was STIRS' first year since the conversion to an API-driven corpus, our system was adjusted to deal with the change from a full corpus to only seeing, at most, 10,000 tweets for each topic, which changed some of our module's strategies.

In order to help deal with only gathering 10,000 tweets for each topic, we employed a technique dubbed "Query Splitting" to find more relevant tweets for each topic. Each query was split into two queries and queried separately from the API. These two separate queries were then combined, in order of each tweet's score, and used in addition to the regular API querying when experimenting with runs. It was found in experiments on the 2011 tweet corpus that this strategy helped when finding relevant tweets for each topic.

3. Query Expansion Modules

Using query expansion is a popular method used in information retrieval. Due to the nature of the API, it could only return the maximum of 10,000 tweets for each topic, and Lucene was only returning those tweets that have any written representation of the current query. The theory is

[5] "Information that would not have been available to the system at the timestamp of the query." (TREC 2014 Guidelines)

[6] A class of tweets that contains the same information.

[7] Apache Lucene™ http://lucene.apache.org/

if each topic was increased with the right words, then the API would be able to find more tweets that are relevant to the topic. The basic idea is to expand on the original query, using a variety of techniques, i.e. looking for synonyms of the query words.

3.1 Google

We noticed that many of the query topics appeared to have very good results when doing a Google search using the full query. For example, one of the original NIST supplied test queries was "*Keith Olbermann new job*." The top Google results mentioned his departure from the cable news network MSNBC and that Olbermann was hired by CurrentTV. We cross-checked these against the tweets in our corpus and noted that valuable terms like *MSNBC*, *fired*, and *CurrentTV* were in many of the tweets. We hypothesized that adding these terms to the original query would help find these tweets. Each query was then googled using our system, where it gathered up the top common words found across links, removed words that are usually found in webpages, and returned the new queries for each topic.

The new rule to the 2014 Microblog track, which forbade groups from using future evidence, was most affected by our system in our Google module. Instead of just randomly googling throughout all possible dates, we utilized Google's advanced search features, which allow users to search for their query between two dates. For each topic, the topic was googled between the beginning date of the corpus and the query date given by TREC for each topic.

3.2 WordNet

One of the most basic query expansion methods is to simply take each individual word and expand it using synonyms and antonyms of the word. We employed Princeton's WordNet[8] encyclopedia, which allows the user to find synonyms, antonyms, and hyponyms of words. For this module, we allowed WordNet to gather the synonyms of each word in the topic, and then returned the synonym that had the same part of speech as the word originally used in the query. We especially found that words that were both considered to be a noun and a verb were better if returning the verb synonym when the word used in a verb manner.

3.3 Links

According to the 2014 microblog track guidelines, NIST assessors would be allowed to follow urls within a tweet and utilize the subsequent web page content to judge whether a tweet was relevant. From our Link Crawling module, we had already established a list of the top thirty urls from within the tweets for each topic. Each of these urls were there scrapped for their content and our system then calculated the common words from among each topic's content, to be re-added as query expansion.

3.4 Internal Query Expansion

For our internal module, we used the initial tweets given by the API and Lucene to extend each of the topics using Lucene's own indexing power. The first step was to send the scrapped API corpus through our slang converter, which found words that were used as slang and converted them to their proper format – for example, "lol" was converted to "laughing out loud", "rofl" to "rolling on the floor laughing", etc. This helps especially in topics that contained acronyms, such as NSA or NIST. Then STIRS created two lists – one list contained all the words per topic, while another list had all the words found throughout the corpus. When processing this algorithm, a threshold was developed for each topic, so the top words would only include those tweets that were actually considered relevant to the topic, as the API had the manner of returning tweets that only had a remote connection to the topic itself.

After the words were gathered, a sorting algorithm was used to determine which of the words would be considered most relevant as well as taking into account the removal of stop words, the removal of words found among other topics, the amount that word was found within that topic's tweets, and whether that word correctly matched its original query's part of speech (using Apache OpenNPL[9]). As this list was determined, a separate list was created, which found unique hashtags amount the topics and returned a separate list of hashtags.

[8] http://wordnet.princeton.edu/

[9] http://opennlp.apache.org/

3.5 Manual Run

The manual run was built in with the query expansion modules; when all query expansion modules were completed for the run, STIRS put together a query for each topic to be re-submitted to the API. If the manual run is selected, the user is able to accept or reject each of the new queries for each topic, adjusting the queries so they're more likely to find tweets related to the topic.

3.6 Query Expansion Future Work

The API naturally uses Lucene to index all the tweets, so when a query is sent through, it simply finds those tweets that precisely match up with the query's words. Each query expansion module so far concentrates on adding onto the original query, but what if the opposite was done, where words that were considered too broad for the topic were removed from the query? It's possible that there would be less of a chance of the API finding tweets not relevant to the topic, and increasing those that are more relevant. For example, the 2011 topic "BBC World Service staff cuts" produced a number of tweets, scored low by Lucene, related to "Nursing staff jobs" – removing the word "staff" would possibly have kept those tweets to a minimum.

When observing WordNet, we've noticed that a combination of noun and verb meant that it was more likely that the verb synonym was more relevant most of the time then the noun synonym, and thus the algorithm always tried to return the verb first. It's possible that other variations of synonym matching exist, and would increase the possibility of WordNet's usefulness.

4. Relevance Modules

Part of the Microblog Track is being able to work with the API to return the best relevant tweets that are possible, while the other half is using various methods to determine whether a tweet is relevant for that topic, and this can be done a variety of ways. For example, many systems take advantage of the urls found within many tweets and judge whether the tweet is relevant based on whether the URL contains useful information on the topic.

4.1 Link Crawling

NIST assessors were allowed to follow URLs within a tweet and utilize the subsequent web page content to judge whether a tweet was relevant. Therefore, we decided to scrape the website content of urls included in each tweet and utilize that information to help judge a tweet's content. After the process of downloading each collection's initial tweet corpus from the API, a Link Crawling was completed on the full collection of tweets returned by the API. Each Link Crawling used the top 1000 tweets, and their urls, returned by Lucene to scrape the content, which was downloaded using web scrapers Jericho[10] and Jsoup[11]. After the content was downloaded, we utilized Lucene to index and search the content from the retrieved hyperlink pages which allowed us to rank each tweet based on how well their hyperlink page scored.

In past years, we had already implemented Lucene to handle indexing and searching our own tweet corpus and discovered that the best way for Link Crawling to work was to combine the search results from the Lucene API baseline with the URL Lucene index. Since the API already uses Lucene for indexing and returning relevant tweets, we were able to continue this experiment. To begin, STIRS goes through the Lucene-made URL index and creates a list of the top thirty urls found within each topic. It then goes through each tweet and adjusts the score, based on the Lucene url scoring, on whether the tweet contains a url, whether it's found within the list, and/or whether it contains multiple urls.

4.2 Machine Learning with WEKA

Without any example topics, when searching for names of politicians, such as Hillary Clinton, there was a trend that the good Tweets tended to be longer and contain urls linking to a news article. This leads to questioning what other "traits" could make a tweet relevant, which eventually established the idea of machine learning utilizing tweet attributes. Machine Learning allows the computer, after being given an algorithm and a training set, to decide whether a tweet is considered relevant.

The attributes selected to train our system this year were: existence of a URL, a hashtag or a mention, number of followers, relevance of URL, number of retweets, maximum retweets, percent

[10] http://jericho.htmlparser.net/docs/index.html
[11] http://jsoup.org/

match between query and tweet, tweets to follower's ratio and relevance. Relevance of urls uses data created by the Link Crawling module to decide if a URL within a tweet is relevant to the query topic. The number of retweets attribute is a numeric value returned via the API, while max retweets is a boolean that is true if the retweet value for a tweet is 100, which is the maximum value returned by the API. The percent match attribute compares the words in each tweet with the words in the current query and returns the percentage of words in common.

After the training set is created, we used Weka[12] to create an algorithm and implement it. Weka, an open source machine learning package developed at the University of Waikato, provides many options for aggregating and viewing data, as well as predicting results for new data given a learning set and selection of a learning model. The final stages of implementation revolved around trying different learning models, i.e. Naïve Bayes, Linear Regression, Decision Trees, etc. and automating the entire process.

This year, we created our own training set based off of the 2013 collection, where 2500 tweets were collected and judged on relevance. For our experiment, the models JRip and J48 were found to be most effective, were correctly classifying at 84.54% and had a 41% precision rate, which meant that our model was judging tweets at about the same precision as Lucene!

4.3 Adjusting Tweets based on Individual Factors

When observing the tweets that the API returned as most relevant for each topic, singular tweet attributes seemed to stand off as indicators that the tweets was relevant. Two of these factors provided success among various experiments: the amount the tweet was retweeted and the percentage of the tweet that contained the original query. In the process, STIRS created a threshold for each topic and then went through the tweet corpus, deciding whether the tweet's score should be adjusted for each of these factors.

After the success of our Google and Link Crawling modules, we created a last relevance module that grabs the top links found when googling each of the top tweets from each topic. If a percentage of the tweets matched up with the query topic's urls, then the tweet's score was re-adjusted depending on the percentage. This module was always run last of any run, and helped with last minute adjustments for score evaluation, particularly for precision@30, among the top tweets.

4.4. Relevance Modules Future Work

For our Link Crawling module, we believe that experimenting with how the content is dissected for analysis might improve the quality at which tweets are considered relevant. Right now, it is purely based off of Lucene's analysis of the content matching each of the topics, but increase precision might occur by introducing reliability metrics, such as the algorithm considering news articles to be of higher relevance then a blog.

Currently, our Link Crawling module also depends on Link Crawling the whole database, before runs are started, in order to find the top thirty urls. However, with the new API set-up, we were unable to provide the Link Crawler every single link that would come up among the tweets, as only the top 10,000 tweets for each topic are available. This comes up as an issue with runs involving query expansion, as Lucene might pick up on the new query and find tweets, especially tweets with urls, that were not available to be Link Crawled, and thus this module loses efficiency.

Using WEKA for Machine Learning allows us to pick, from among a variety, a classifier to use for choosing which tweets were considered relevant. However, our program currently only uses one classifier to decide – a different scheme could be used by having multiple classifiers in use, and have them "vote" on whether a tweet was relevant.

5. Tweet Timeline Generation Module

Although our prime focus for this year's Microblog track was towards the adhoc task, we devoted some time to creating a somewhat credible run for the TTG task. The STIRS system would naturally use our best adhoc run (the run that had the highest precision@30) as the relevant tweets for consideration in the TTG run. The module went through the top one hundred tweets

[12] http://www.cs.waikato.ac.nz/ml/weka/

for each topic and calculated the percentage of the tweet compared to that of each cluster grouping. If the percentage was higher than the given threshold, the tweet was considered part of that cluster group, but if it went through all the cluster groups and did not match any, it was considered a new cluster group. Each group of clusters was separated by topic, and each topic printed the first tweet discovered for each cluster group. Scoring and rank was not considered, nor whether the first tweet represented the best tweet for that cluster.

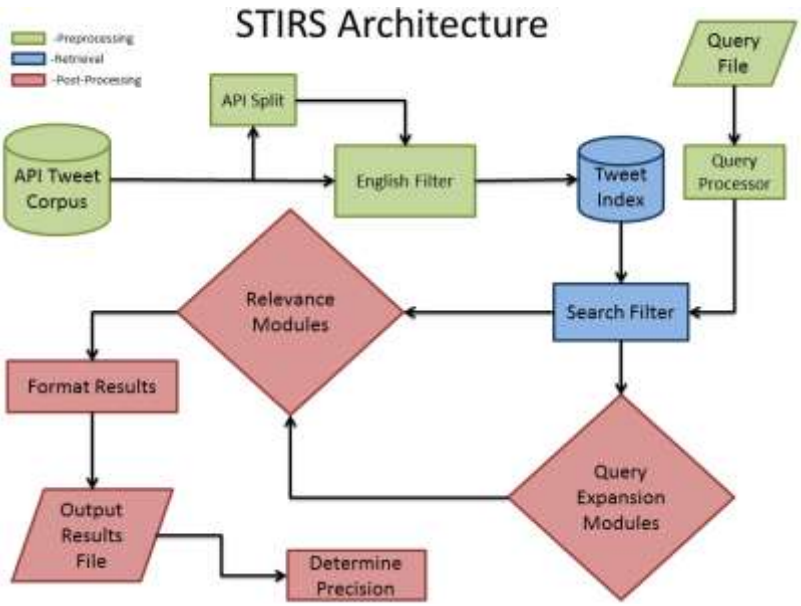

Figure 1: STIRS System Architecture Diagram

6. STIRS

We incorporated all of our twitter modules with other necessary modules, i.e. Query Processor, Lucene Processor, TREC formatter etc., into a fully automated end-to-end STIRS system (Figure 1). Our Query Processor module converted the TREC formatted queries into Lucene format. Our Lucene processor module returned a Ranked Tweet List (RTL) for a given input query. The TREC formatter converted our RTLs into the standard TREC format, using a Normalizer to collapse all scores on an equivalent threshold. STIRS was developed such that any given module could be easily turned on or off to allow for multiple combinations of experiments, i.e. Query Expansion → Link Crawling: run the query expansion module followed by the link crawling module.

7. STIRS Submission

We experimented with all possible combination of our TM modules on the example topics, in order to select the four best combinations to send to NIST for evaluation. Judgments were made by all team members and were done on a relevant/non-relevant basis for each tweet.

Our four highest performing modules were:

1. Manual CommonWords Query Expansion with Adjusting Tweets Based on Individual Factors
2. Manual Google Query Expansion with Link Crawling and Adjusting Tweets Based on Individual Factors
3. Automatic Run with Link Crawling and Adjusting Tweets Based on Individual Factors
4. Automatic Run with Link Crawling, Machine Learning and Adjusting Tweets Based on Individual Factors

We selected these four versions of the system to run on the official topics and return to NIST for evaluation.

8. Official NIST Results

The judging showed our average to be at 61.91% precision for the Adhoc Task, with an average of 85.38% precision for the Tweet Timeline

Generation Task. Both precisions estimated at 30.

8.1 NIST Task Scoring Metrics

For the 2012 track NIST used different scoring metrics for the adhoc task and the tweet timeline generation task. The adhoc task had three scoring metrics. These were ranked based on precision, ROC curve, and recall. However, since there was no single summary value for the ROC measure we will only report precision@30 in our results for this track.

The TTG task had two different scoring metrics: cluster precision and cluster recall, which would be calculated in both an unweighted and weighted version. Cluster precision indicated how many distinct cluster groups were present for each topic, while cluster recall is how many of the cluster groups that were discovered by TREC are present in the run. Cluster precision and cluster recall will be combined into the F1 metric to receive a score for the run:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$ [13]

The unweighted version does both these scoring metrics as discussed, but the weighted version takes into account the fact that some cluster groups are more relevant than others, and will affect the score appropriately. This means that each run will receive two scores for each version.

9. APPENDIX

9.1 Sample Query

<top>
<num> Number: MB001 </num>
<query> BBC World Service staff cuts </query>
<querytime> Tue Feb 08 12:30:27 +0000 2011 </querytime>
<querytweettime> 34952194402811904 </querytweettime>
</top>

9.2 Sample Tweet (API)

MB001 Q0 29983478363717633 1 5.316302 myRun# TResult(id:29983478363717633, rsv:5.3163018226623535, screen_name:fatima9632, epoch:1295983592, text:[BBC News] Major cuts to BBC World Service: BBC World Service is to close five of its language services, with th... http://bbc.in/e2vlpX, followers_count:1, statuses_count:13794, lang:null, in_reply_to_status_id:0, in_reply_to_user_id:0, retweeted_status_id:0, retweeted_user_id:0, retweeted_count:0)

9.3 Sample Submission

MB01 Q0 3857291841983981 1 1.999 myRun

10. ACKNOWLEDGEMENTS

We'd like to thank The Siena College Institute for Artificial Intelligence, the Center for Undergraduate Research and Creative Activities, and all previous STIRS researchers.

11. REFERENCES

1. http://jericho.htmlparser.net/docs/index.html
2. http://jsoup.org/
3. http://lucene.apache.org/
4. http://opennlp.apache.org/
5. https://twitter.com/
6. http://www.cs.waikato.ac.nz/ml/weka/
7. http://wordnet.princeton.edu/
8. https://github.com/lintool/twitter-tools/wiki/TREC-2014-Track-Guidelines
9. https://github.com/lintool/twitter-tools/wiki/TREC-2013-API-Specifications

[13] http://en.wikipedia.org/wiki/F1_score