10 Weeks to TREC: STIRS
Siena's Twitter Information Retrieval System

Sharon Gower Small, Darren Lim,
Karl Appel, Denis Kalic, Matthew Kemmer, David Purcell, Carl Tompkins, Chan Tran
The Siena College Institute for Artificial Intelligence
Siena College
Loudonville, NY 12211
{ssmall, dlim, km25appe, d30kali, me21kemm, da14purc, cl05tomp, cs05tran}@siena.edu

ABSTRACT

There has been an increasing interest, both of the research community and federal funding agencies in microblogs as a source of viable information for a variety of tasks. NIST (National Institute of Standards and Technology) has added a microblog retrieval track to TREC (Text REtrieval Conference) for the first time in 2011. NIST has selected Twitter as the source of microblog data. Twitter is a dynamic social website that allows users to post tweets which are short posts to share news with friends and followers across the world. While some tweets provide useful information, this information is very limited by the restriction on length to 140 characters or less. Participating teams were provided with the code necessary to download the Twitter Corpus, consisting of 16,141,812 tweets from a 2-week time period, January 24, 2011 to February 8, 2011, inclusive. Teams were also provided with a training set of 12 example topics, and later the test set of 50 topics. In this paper, we describe three modules designed for this track, built within a system called STIRS, Siena's Twitter Information Retrieval System. After submitting three user-defined runs and a Lucene baseline run, the NIST judging showed our best run to be at 30.83% precision. The reported median from all runs of all 58 participating teams was 25.9%. We also describe our process of developing a new and complete end-to-end system in just 10 weeks time with six undergraduate researchers.

General Terms
Algorithms, Experimentation

Keywords
Twitter, TREC 2011, Microblog, Lucene, Weka

1. INTRODUCTION

In May 2011, members of the Siena College community formed a group dedicated to working on the TREC Microblog competition. The group was lead by Sharon Small, a long-term researcher in the field of Computational Linguistics as well as a pervious participant in the TREC QA track. The team was comprised of a fellow colleague of hers, Darren Lim, and six undergraduate researchers: Karl Appel, Denis Kalic, Matthew Kemmer, David Purcell, Carl Tompkins and Chan Tran. Starting in the middle of May, the group began work on STIRS.

1.1 Information on TREC Microblog

The TREC microblog track is a new edition to the TREC tracks for 2011. The corpus for this track is a 16,141,812 tweet collection obtained from a two week period, January 24, 2011 to February 8, 2011, inclusive. Queries on this corpus took on a particular form, where "users" wanted to find up to date and relevant information about a news topic. In addition to a short stated topic of interest, e.g. "*State of the Union and social media*,"[1] each query also contained a query time. Information returned by the systems could not be older than the moment of the query time or it would be judged as irrelevant. For each of the 50 test topics a system needed to return a time ordered set of tweets; the NIST assessors would judge the first 30 tweets.

Each participating group was allowed to submit up to four different runs, where each run consisted of a set of ordered tweets for all 50 topics. One of the runs was required to not utilize any outside

[1] The xml format of the topic file used by NIST may be seen in Appendix A.

information, i.e. future tweets, web mined information, etc.

The corpus and example topics were released May 20, 2011, with all runs from participating teams due to NIST by August 11, 2011. Official results were released in late September 2011.

1.2 The System

Our team utilized an 8-processor, 64-bit Dell Precision 490 for downloading the corpus, developing STIRS and executing various experiments. Each processor is an Intel Xeon 3.00 GHz CPU, each having a two CPU core. This server has 16 GB of memory and 2.25 TB of hard drive space. It is running Redhat Linux Enterprise Version 4.

1.3 Our 10-week Trek to TREC begins

The timeline of our ten weeks consisted of a kick-off meeting between Dr. Small and the rest of the team two weeks prior to the official start of the student researchers' work period. Soon after Siena's classes ended, the team members commenced work on STIRS. The team met on a weekly basis (sometimes as many as three times in a week) to discuss progress on the system.

Preliminary Week 1: May 15-21: Conceptual system design of STIRS was completed during our first preliminary week. We envisioned an agile system that incorporated processing modules, which could interact with each other or by themselves, to support a variety of different experiments on the microblog data.

In order for the group to proceed with the development of STIRS, several organizational issues needed to be addressed. Firstly, we selected Lucene[2] to index our tweets corpus, and we successfully installed and configured Lucene on our server. The undergraduate researchers were then split into teams of two, formed for the purposes of each generating an individual Twitter Module (TM) that would potentially increase our precision. Each module was required to work completely standalone within the STIRS system, or as input and/or output modules to other TMs as applicable.

On May 16, 2011, the downloadable materials (tweet ids and scripts) of the microblog corpus were made available by the organizers. After readying the scripts on our machine, the download process[3] was started on May 20th. Our initial download speed of this single process was extremely slow (1 .dat file[4] pair processed every twenty minutes), which would have taken us far past our initial deadlines. We quickly ran some experiments to determine how many parallel processes we could spawn on our server before degradation in performance occurred. These experiments showed that four processes could successfully be run in parallel to speed up our download process without any degradation in performance. We began to modify our code to spawn multiple processes to download the entire corpus without repetition.

Preliminary Week 2: *May 22-28*: We established our communication resources, including shared work directories on our server, a Google sites/calendar location and a temporary "course" on our school's Blackboard server. This facilitated our resource organization.

In the meantime, other parts of STIRS were being developed. Code that would process TREC queries into input queries for Lucene was developed at this time, as well as code that would take Lucene output and generate TREC-format results for submission purposes. Also during this week git[5], a version control system was installed on our server.

Week 1: May 29-Jun 4: The 4-process download module was ready and started on May 29th. The downloading of the corpus was completed on June 2nd, and Lucene indexing immediately commenced. During this time, the module teams continued to propose and refine ideas for their modules. Additionally, the Query processor and TREC formatter modules were completed. Other parts of our system were also being developed, such as a textese component, which would transform Twitter slang into its full English counterpart. Given the 140 character length restriction per individual tweets, a great deal of

[2] Apache Lucene™ is an open source high-performance, full-featured text search engine.

[3] Teams were only supplied with Tweet ids, not the actual text of the tweet. Teams were required to mine Twitter to obtain the actual text of the tweets in the corpus.

[4] NIST provided 1673 .dat files, each containing a block of tweets ids to be downloaded

[5] git is a free and open source, distributed version control system.

textese, i.e.: g2g = got to go, is used by tweeters. The same textese was not used in the example topics and therefore we needed a conversion module.

Week 2: Jun 5-Jun 11: According to the microblog guidelines only English tweets would be judged as relevant by NIST assessors. Therefore, we developed a simple foreign language filter. This filter reduced our ~16 million tweets corpus to 5,448,156. On week 2, we met our deadline of having a complete indexed Twitter corpus by the first half of June. We then developed a simple script to query the Lucene index directly in a terminal window; this allowed for quicker prototyping of ideas and strategies for modules.

On this second week, teams also finalized their ideas for their Twitter modules. Section 2, 3, & 4 below will give a detailed description of each of the three TMs. Section 5 of this paper will give a high level overview of the entire STIRS system. Finally, we will report on our team's NIST results.

2. TM 1: Scraping URLs

2.1 TM1 Motivation

According to the micro-blog 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 or not. Given this, student researcher Team One proposed to use information from the URL links present in a tweet to determine whether a particular tweet was relevant. Throughout the course of this research, we utilized the 5.5 million English tweet corpus. Of these 5.5 million tweets, we automatically detected that approximately 1.3 million contained hyperlinks. Of those hyperlinks, 1.1 million hyperlink pages were able to be downloaded using web scrapers called Jericho(1) and Jsoup(2). After completing the download, we utilized Lucene to index and then search the content from the retrieved hyperlink pages. Each tweet could then be ranked based on how well their hyperlink page scored. This first approach only ranked tweets that actually contained a URL.

2.2 TM1 Expansion

After error analysis on the initial results, we attempted two modifications to further improve the precision of our retrieved tweets. The first was to give a penalty to content that did not contain all the query terms or associated synonyms. The second was to penalize web pages with fewer words and to give a bonus to those with a higher word count. These modifications were inspired by results retrieved from the initial run of the query examples on our corpus. For example, the first result for "Australia natural disasters" was for the ten word home page of a natural body building site in Australia, or when searching for the terms "Chavez expropriate property" we got an adequate number of results dealing with people who had the last name of Chavez but nothing associated with Hugo Chavez. From our error analysis of this initial run we determined that in general, content with a higher word count were typically more reliable pages. This first approach only ranked tweets that contained a URL.

A separate approach to improve results was the consideration of both tweets with URLs and tweets without URLs. To begin the process, two additional Lucene searches were performed on two different indices. The first index was built using the text of the tweets themselves. The second index was built using the text from the web pages linked in the tweets. Each of these searches created a ranked list of tweet results. We attempted to improve results by merging these two lists into a single list by using a process we refer to as a "ranked join." The first step in the merge was to normalize the scores of both lists. All unique tweets were then taken from both lists and put into a single list in no particular order. Each tweet in the new list was then processed and assigned a new score. We experimented with several scoring modifiers:

- If a tweet contained multiple URLs, its score was increased by the sum of its URL scores.
- If a tweet occurred in both of the original lists, its score was increased by an intersection bonus value.
- If a tweet contained no URL its score was decreased by a penalty value.

Most of the development of this approach was focused on adjusting the scoring modifiers through iterative testing

2.3 TM1 Future Work

We were limited in our time, just 10 short weeks, as well as the lack of a judged corpus. We plan to train our module for improved performance utilizing the newly released TREC judged micro-blog corpus. This corpus will allow us to train for a variety of weights and thresholds: i.e. ideal penalty scores, determine the best formula for combining the Lucene tweet results and the Lucene web page results, etc. Additionally, as this module utilizes linked URLs, we believe we may be able to further increase precision by utilizing some reliability metrics for the web pages linked to, e.g. news article on cnn.com may be more reliable than those on a blogging site.

3. TM 2: Feature Modeling with WEKA

3.1 TM2 Motivation

The motivation for this module came about while examining the corpus. Without any example topics, Team Two searched for names of politicians, such as Ron Paul and Hillary Clinton. Looking at the results, we noticed a trend that the good Tweets tended to be longer and contain URLs linking to a news article. This lead us to question what other "traits" of tweets could make them relevant, which eventually lead to the idea of machine learning utilizing tweet attributes.

3.2 TM2 Method

The majority of time spent implementing TM2 dealt with the Twitter API to download data about the users in our corpus. Earlier in week 3, we registered STIRS on Twitter in order to be allowed to use the Twitter API. Ultimately, the package used for implementation was Twitter4J, a Java library for the Twitter API.

Through trial and error and a review of relevant research, several attributes were chosen to be the ones used for feature modeling. They were as follows:

- Tweet length – the number of characters in the tweet
- URL existence – whether a tweet contains a URL or not
- Follower Count – the number of followers a user has
- Friend Count – the number of other users a particular user follows
- List Count – the number of Lists[6] on Twitter a user appears on

The first two attributes were computed for each Tweet in the corpus, while the next three were downloaded from the Twitter API for each user in the corpus. The download consisted of 2,237,031 users who both posted in English and had active accounts at the time of download. The process ran between July 2nd and July 5th, having to abide by Twitter's cap of 350 API requests per hour. The means for accessing the API is a simple registration process to receive authorization codes, which then must be presented when making API requests. To receive this particular information, all that's needed is the desired username, and all of their information which is publicly accessible on Twitter.com may be downloaded.

It wasn't until the final weeks that we were ready to utilize Weka, an open source machine learning package developed at the University of Waikato, New Zealand. Weka 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.

The verification process alternated between judging our results as to whether we felt that they were relevant or not, and generating new machine learning models from our test data. 10-fold cross-validation was used for training and testing purposes. Through our testing process, some attributes, such as expert ratings (calculated by using data from the site Listorius.com), did not prove to be useful. Judgments were made by all team members and were done on a relevant/non-relevant basis for each graded tweet. Our results proved to be moderately successful, though not as good as we had hoped.

3.3 TM2 Future Work

While the initial results showed an improvement over the baseline results, the combinations with other modules consistently produced better results

[6] Specifically, a List on Twitter is a sub-group of followers that may be further categorized to allow for easier viewing.

than this module on its own. We feel we were severely limited by the size of our training corpus. Given our time restrictions we were only able to generate judgments for our 33 example topics, 100 tweets each. We plan to utilize the much larger TREC judged corpus to train our system and hopefully achieve an improved precision.

4. TM 3: Query Expansion

4.1 TM3 Introduction

Initially, Team Three approached their module design with query expansion in mind. We examined query expansion by traditional successful techniques, i.e. looking for the synonyms of the query words. They used Princeton's WordNet, which allows the user to find synonyms, antonyms, and hyponyms of words. For each query, they used WordNet to expand the query by adding its synonyms. A second approach utilized Wikipedia search capabilities to find relevant articles for each word/phrase within the query. Once our program found a Wikipedia article, it used the first page it found to find the most commonly occurring words within the page. By using this methodology, the most commonly occurring words and phrases (after eliminating stop words) were utilized for query expansion terms.

When testing TM3, we experimented with four configurations: WordNet only, Wikipedia only, WordNet & Wikipedia, and Wikipedia & WordNet. We combined Wikipedia and WordNet in a linear fashion: we first expanded our query using either Wikipedia or WordNet, we then took that expanded query and sent it to the opposite query expander, expanding further on the query.

4.2 TM3 a New Approach

After our results for Wikipedia and WordNet, we decided to quickly change our approach to the query expansion problem. At first, we attempted to use Google News[7] for query expansion. We decided to experiment with Google News because the test topics supplied by NIST were primarily related to current news stories of interest. However, we quickly found that due to the limit on the Google news archive, we could not find many stories that were relevant to the query. However, for the results that Google News did find, the articles were very relevant to the topic, including exact phrase matching in some cases. This discovery illustrated the power of using Google search results for query expansion. We quickly switched to Google for query expansion and found that, on average, the top four results produced the most pertinent pages. We decided to fetch these pages, unless the URL contained the word "video" or "youtube.com" as we found those pages typically did not offer relevant terms. Our module then detected the most common words/phrases in the text. By comparing this list of words between all of the pages, we retained the four most reoccurring words/phrases and added these to the query. We chose to use four words since the fifth word tended to pick up more tweets that were irrelevant when experimenting with the NIST supplied test topics.

We found our Google module results to be much better than WordNet and Wikipedia results. Five of our queries came out unusable in which the html code of the website was being returned was mistakenly added to the query. This expanded our query with irrelevant information that did not help our results. Even so, the results were not affected heavily by this bad query. Thirty-four of our results were changed and expanded while eleven remained unchanged. The expanded queries showed moderate improvements to the results.

4.3 TM3 Future Work

From this research, we found that query expansion through Google saw a greater increase in precision than our WordNet and Wikipedia classes. Though there were flaws such as HTML code being returned in some queries, our Google query expansion module worked far better than utilizing the synonyms of WordNet or common words of Wikipedia. We have illustrated that query expansion utilizing Google can improve the precision of our Twitter results. We plan to run additional experiments with the NIST judged microblog corpus to verify and/or improve our module: e.g. is 4 pages the best depth? what is the best number of words/phrases to add to our expanded query, could these thresholds be topical or related to the types of pages being returned, etc.

[7] http://news.google.com/

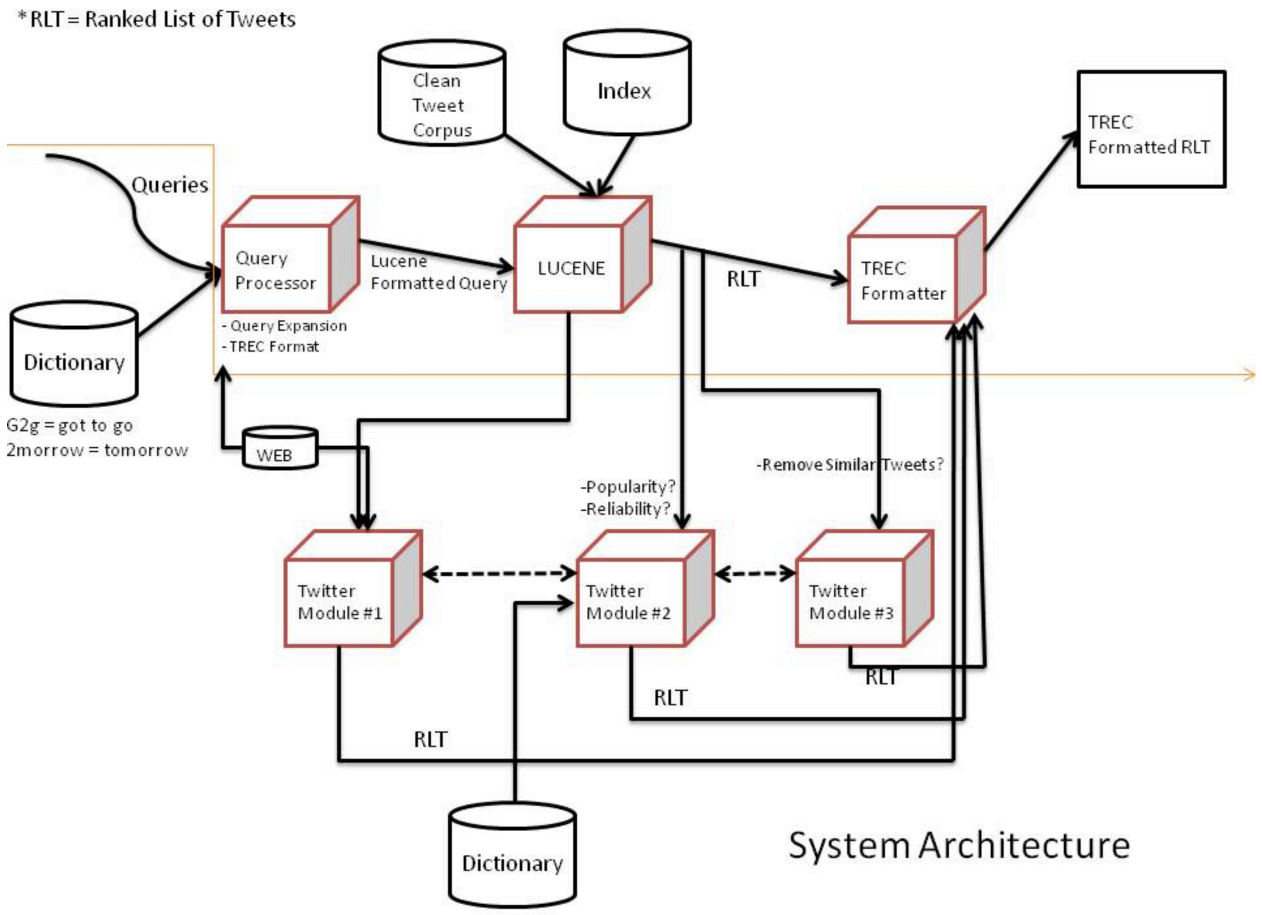

Figure 1: STIRS System Architecture Diagram

5. STIRS

We incorporated all three 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 XML formatted queries into Lucene format. Our Lucene processor module returned a Ranked List of Tweets (RLT) for a given input query. The TREC formatter converted our RLTs into the standard TREC format. STIRS was developed such that any given module could be easily turned on or off to allow for multiple combinations of experiments, i.e. TM3 -> TM1: run the query expansion module followed by the URL ranking module.

6. End-to-End STIRS Experiments

Once each team felt they had the best version of their module given the allowed time, full end-to-end system experiments began.

We experimented with all possible combination of our TM modules on the example topics, in order to select the 3 best combinations to send to NIST for evaluation (one run sent would be our baseline run to fulfill the requirement of no outside resources utilized). Judgments were made by all team members and were done on a relevant/non-relevant basis for each tweet. We scored the top 30 tweets for each of the 33 example topics[8] where each tweet was scored by at least two judges.

[8] We expanded on the 12 example topic set supplied by NIST for a total of 33 topics for testing purposes. See section 9.2

Our highest performing modules were:

1) TM3 -> TM1; Query expansion followed by our module that utilized urls within the tweets

2) TM1 alone

3) TM3 -> TM1-> TM2; Query expansion followed by the url modules, followed by the Weka module

We selected these three versions of the system to run on the 50 test topics and return to NIST for evaluation.

7. Official NIST Results

The judging showed our best run to be at 30.83% precision. The reported median from all runs of all 58 participating teams was 25.9%.

8. STIRS Conclusions and Future Work

We were able to build a fully functional STIRS system in just 10 short weeks that performed well above the median reported for the microblog track. We look forward to making significant progress on our system, as described in our module sections above, now that we have the valuable NIST judged microblog corpus. We plan to report improved results in our full TREC proceedings paper.

9. APPENDIX

9.1 Sample Query

```
<top>
<num> Number: MB01 </num>
<title> Wael Ghonim </title>
<querytime> 25th February 2011 04:00:00 +0000
</querytime>
<querytweettime> 3857291841983981
</querytweettime>
</top>
```

9.2 Experiment One Test Topics

The first twelve were supplied by the TREC organizers. The rest were developed by Dr. Lim using back issues of the New York Times to find "appropriate" news stories which were similar in stature to previous queries.

- Chavez expropriate property
- Jintao visit US
- Saleh Yemen overthrow
- Sudan independence vote
- natural disasters Australia
- Kepler discovers new planets
- Texas school robot
- State of the Union and social media
- Cavaliers record
- Sian Massey comments
- Mets, Madoff victims lawsuit
- Bjorn Qatar Masters
- Groundhog Day Celebration
- Urban Meyer ESPN
- NHL Choosing Sides
- News of the World Hacking
- Columbia Coal Mine Deaths
- Kucinich lawsuit
- Nielsen IPO
- Oscar nomination snubs
- Chrysler quarterly loss
- Belgium Government protests
- Jack LaLanne death
- Loughner plea
- Russian Airport suicide bomber
- Connecticut donation return
- The Kennedys Reelz
- Ireland Russian diplomat
- Myanmar President Candidates
- Sabres sale agreement
- Hydrofracking diesel fuel
- Apple e-book purchases
- Fernando Torres Chelsea

10. ACKNOWLEDGMENTS

This research project was sponsored by the Siena College Summer Scholars Fund.

11. REFERENCES

[1] http://jericho.htmlparser.net/docs/index.html

[2] http://jsoup.org/

[3] http://lucene.apache.org/

[4] http://www.wikipedia.org/

[5] http://twitter4j.org/en/index.html

[6] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[7] George A. Miller (1995); WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.