Siena's Twitter Information Retrieval System: The 2012 Microblog Track

Karl Appel, Lauren Mathews
Darren Lim & Sharon Small
The Siena College Institute for Artificial Intelligence
Siena College
Loudonville, NY 12211
{km25appe, li08math, dlim, ssmall} @siena.edu

ABSTRACT

Since 1992, the National Institute of Standards and Technology (NIST) has been annually hosting the Text Retrieval Conference (TREC). One of the newest tracks, which started in 2011, is the Microblog Track, which uses a well-known social network site, Twitter[10], as its source of microblog data. Twitter allows its users to post 140 character length tweets to share messages with their followers, posting personal updates, and share major media stories from around the world. In order to evaluate information retrieval on microblog data, groups were provided with a file of about 16 million tweet IDs from January 24th to February 8th, 2011. This allowed us to download the tweet content of each ID for a total of 16,141,812 tweets. Participating teams were given a set of topics to test their retrieval process, and their program would return relevant tweets about that topic. The Siena College Institute of Artificial Intelligence expanded on STIRS, Siena's Twitter Information Retrieval System. The results for our adhoc run showed STIRS' best run to be at 18.08% precision, while the average of the median from all participating teams was 14.86%.

1. The Microblog Track

Continuing into its second year, TREC's Microblog Track continues to be one of the most popular tracks. Twitter, a social networking site, provided a 2011 tweet corpus[1]. This was reused for the 2012 runs and it consists of 16,141,812 tweets obtained from January 24th to February 8th, 2011. The tweet information we downloaded included user ids, dates, query times and the actual tweet content. NIST also provided a set of test topics that would be used as input queries to a participating team's system. The goal was to return only tweets relevant to the query 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. Groups submitted up to four different runs, each run consisting of a set of ordered tweets for each of the fifty topics. Different from the 2011 task, this year's runs were divided into two distinct categories: filtering and adhoc.

[1] NIST provided teams with a list of tweet ids and the tools needed to download the text of the tweets. Actual tweets were not provided.

1.1 Adhoc Task

The adhoc task was similar to the Microblog 2011 task. Participants were asked to retrieve results for 60 new queries. For each query, participants were asked to send back the top 10,000 tweet results they found. However, only the top 1000 tweets for each query were to be judged. For each result there were four fields required to be submitted to NIST. These four fields were the topic number, tweetID, score, and run name. Each judged tweet would be ranked as either relevant or non-relevant.

1.2 Filtering Task

The filtering task was completely new to participants of this year's track. This task reused the 49 topics that were provided by NIST in the 2011 Microblog Track. For this track each participant was provided with two time stamps: the earliest relevant tweet found and the oldest relevant tweet found as judged during the 2011 Microblog Track. Participants needed to return tweets between these two time stamps. In addition, we were asked to submit whether the system determined a tweet to be relevant or not. This was either a "yes" response which indicated a tweet to be relevant or a "no" response which indicated a tweet not to be relevant. Overall there were 5 fields for each result submitted. These were the topic number, tweetID, score, decision of relevance, and the run tag. Only "yes" tweets were considered in scoring results. Of the 49 topics judged last year, NIST set aside 20% for training data. The other 80% of the data was sent for scoring.

1.3 The System

Our team utilized an 8 core 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.

2. STIRS Module One: Scraping URLs

2.1 Motivation

NIST assessors were allowed to follow URLs within a tweet and utilize the subsequent web page content to judge whether a tweet was relevant or not. Therefore, we decided to scrape the website content of URLs included in each tweet and utilize that content to help judge whether a tweet was relevant or not. Throughout the course of this research, we utilized the 5.5 million English tweet corpus[2]. Of these 5.5 million tweets, we automatically detected that approximately 1.3 million contained hyperlinks utilizing regular expressions. Of those hyperlinks, 1.1 million hyperlink pages[3] were able to be downloaded using web scrapers called Jericho(1) and Jsoup(2). We utilized Lucene[3] to index and then search the content from the retrieved hyperlink pages which allowed us to rank each tweet based on how well their hyperlink page scored.

[2] Non- English tweets were judge irrelevant by NIST, therefore we filtered non-english tweets from our corpus.

[3] .2 million were "dead" links – no page found, etc.

2.2 Method

After analysis of our results and continued experiments from last year we saw that a combination of both information from our baseline Lucene results and information from URL pages were best at retrieving relevant tweets. To begin the process, two Lucene searches were performed on the two different indices. The first index used was built using the text of the tweets themselves. The second index was built using the text from the web pages linked in tweets. Each of these searches created a ranked list of tweet results. We improved 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 merged into a single list based on their normalized scores. If there were duplicates, meaning the same tweet was in both list, the tweet score from the URL list was used.

3. STIRS Module 2: Feature Modeling with WEKA

3.1 Motivation

Utilizing our work done for the 2011 Microblog track, we continued experimentation with machine learning models. We modified our approach slightly utilizing a different set of attributes to learn on. i.e. adding part-of-speech information. We tagged each tweet using Apache OpenNPL[9]. The four parts of speech we used for our model were nouns, verbs, adjectives, and adverbs.

3.2 Method

For this year's task we used the 2011 judged set provided to us by TREC for training and testing our models. In total there were ~60,000 judged tweets from last year's track where over 2000 tweets were judged as relevant or highly relevant[4]. For our experiment we created a set of data that would contain half relevant tweets and half non-relevant tweets. Once the data was properly formatted and our set was created we proceeded to extract the attributes of the tweets.

3.3 Preparing the Data

We included a wide range of attributes in our learning experiments. The attribute values were either represented as a boolean, a percentage, or a whole integer. Our following attributes and their categories are listed below:

Boolean Values (representing existence of the item):

1. URL
2. Hashtag
3. Asterisks
4. Emoticons

Percentage Values:

1. Number of Capital Letters vs Total number of letters
2. Number of Vowels vs Total number of letters

[4] The 2011 Microblog Track had three possible judgments: not relevant, relevant and highly relelvant.

3. Number of Constants vs Total number of letters
4. Numbers vs total number of words
5. Noun vs total number of words
6. Verb vs total number of words
7. Adjective vs total number of words
8. Adverbs vs total number of words

Integer Values:

1. Number of Question marks and Exclamation Points
2. Length: Number of characters in the tweet

3.4 Experimentation:

For our experiments we tried different power sets of the traits we had listed in the prior section on a Bayes Network Model. We used 10 fold cross when testing our data. After testing we discovered that the most effective data to use was the URL, Hashtag, Asterisk, Emoticons, number of question marks and exclamation marks, and the percentage of a tweet that contains a number. Overall we were able to classify 64.9% tweets correctly with these characteristics.

4. Query Expansion

4.1 Introduction

Using query expansion is a popular method used in information retrieval. The basic idea is to expand on the original query, using a variety of techniques, i.e looking for synonyms of the query words. One of our STIRS query expansion modules took each individual query term, omitted stopwords, and found its Wikipedia page. Finding a corresponding Wikipedia page for each word is as simple as concatenating the word to the end of a Wikipedia URL. For example, given the query term "golf," the Wikipedia target page URL would be http://en.wikipedia.org/wiki/golf.

Using this technique would net favorable results, unless a disambiguation page was found because the word was not unique enough to redirect to a single page. A disambiguation page is a Wikipedia page that lists possible topics related to a user's search if one specific topic cannot be found for the user's search. If a disambiguation page was found instead of a single topic page, we could use the disambiguation page itself for query expansion; instead of choosing a potentially erroneous page from the disambiguation page. The text found on the disambiguation page could be used for query expansion the same way a regular Wikipedia topic page would be used. The text on disambiguation pages contain short summaries of each term and contain terms relevant to the topic. One of the terms on the disambiguation page is the correct topic, and overlap of related terms from other topics would help select the proper tweets. Once each Wikipedia page for each query term is found and tags are stripped, the top four most common words/phrases on each individual page, excluding stop words, are compiled into a list. We then compare each of these lists from the unique pages to all the other page's lists, looking for the top four most common phrases between the lists. If there is a tie for the number of times a word appears between the pages and four common words have not already been found, the word that appears earlier on the page that was processed first is taken. The four words are then appended to the end of the query. An example query and its expansion would be:

"Tiger Woods PGA win" =>"Tiger Woods PGA win golf tournament Masters victory."

We also experimented with a second technique for query expansion. We noticed that many of the test query topics appeared to have very good results when doing a Google search using the full query. For example, one of the 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. We implemented a module for these pages similar to our Wikipedia module.

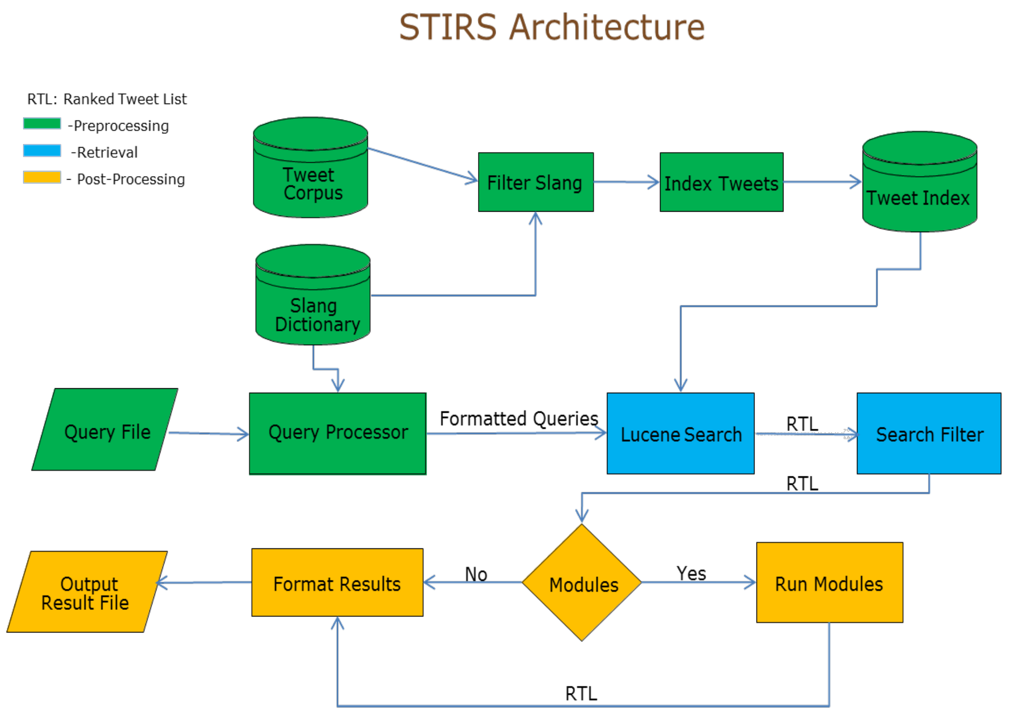

Figure 1: STIRS System Architecture Diagram

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 Tweet List (RTL) for a given input query. The TREC formatter converted our RTLs 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. QueryExpansion -> URLRetrieval: run the query expansion module followed by the URL ranking module.

6. STIRS Submission

For the adhoc task we submitted the baseline results for Lucene and one for each of the three modules we had created. The following is the list of the modules we submitted from highest precision to lowest precision:

1) Baseline: Raw Lucene score based on tweet content
2) TM1: Our module that utilized URLs within the tweets and baseline results
3) TM2: Machine Learning Module
4) TM3: Internal Query Expansion

7. Filtering Task Experiments

We chose to evaluate tweets and score them based upon Lucene's real-time indexing. For each topic a relevance score was first computed using our baseline Lucene run. Our RankedJoin run and our Machine Learning run utilized the baseline scores to generate their final results. Our URL run used only the scores from the index built on our scraped web pages.

We ran three experiments with our different modules based on several hypotheses. Our first experiment was with our RankedJoin module. For this module, 50% of the score was based on how well the tweet did on its baseline run while 50% of the tweet's score was based on how well it did based on the URL information. We performed our experiment this way because our hypothesis was that information from both sources were equally important and should be valued the same. Therefore, each factor would be given equal scoring strength. If a tweet did not have a URL, its URL score would be zero. For our machine learning experiment we generated a Bayes Net model using Weka. Anything ranked under 0.5 by our Bayes Net model was considered irrelevant. This threshold was determined based on manual analysis of test runs. All of our baseline scores were then increased by the value of the Bayes Net score that was over our threshold of 0.5. This run was based on the hypothesis that the relevance score of a tweet generated from our learning model would be a positive factor to determine a tweet's overall relevance. Our URL only module was based solely upon the index built on our scraped web pages. This experiment was based on our final hypothesis, that information from URLs alone could predict relevance.

The Filtering Task required us to indicate our system's belief as to whether a tweet was relevant or not. Each tweet needed to be tagged with a "yes" or a "no" to indicate our system's relevance judgment. Through manual analysis on our test results we discovered a threshold of 1.0 worked best to indicate the relevance of a tweet. Therefore, if the tweet score was above our threshold we would give a relevance decision of "yes". Any other scores would be given a decision of "no".

7.1 NIST Filtering Task Scoring Metrics

For the 2012 track NIST used different scoring metrics for the filtering task and the adhoc 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 filtering task had 4 different scoring metrics. All the metrics were based on the pool of "yes" tweets submitted. The metrics were precision, recall, F-Score, and the T11SU utility measurement. A reference to how these measures were used in the past can be found in the paper "The TREC 2002 Filtering Track Report" [8].

8. Official NIST Results

The official scoring for our filtering run showed our best score to have a T11SU of 35.88%. However due to an error or in our algorithm, we needed to rerun and rescore our filtering modules. We generated our new results and scored these using the filtering script provided to us by NIST. We discovered that our URLOnly module performed the best. The URLOnly module had a precision of 22.81%, recall of 44.36%, F-Score of 20.41%, and T11SU of 15.42%. Even though several other modules had higher recall, our URLOnly module showed the highest T11SU, which is an indicator of how useful our system might be to a potential user. The reported average of the median from all runs from the other participating teams showed precision of 17.6%, recall of 33.4%, F-Score of 35.70%, and T11SU of 20.7%. Our F-Score was not significantly different from the median while our recall and precision were significantly above the average.

The judging for the adhoc run showed our best run to be 18.08% for precision@30. This was from the RankedJoin list module. The reported average of the median from all runs from the participating teams was 14.86%. Our run did 22% better than this average.

9. Future Work

One of our future goals is to experiment with integrating our modules together to improve our results. These modules are machine learning, query expansion, and our RankedJoin list. We believe that better results may be achieved by generating the best results of each module and then combining them effectively together. We plan to experiment more and combine the modules together more efficiently to boost results. In addition we wish to do more work on improving our query expansion.

10. APPENDIX

10.1 Sample Query(Adhoc Task)

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

<querytweettime> 3857291841983981 </querytweettime>
</top>

10.2 Sample Query (Filtering Task)

<top>
<num> Number: MB01 </num>
<title> Wael Ghonim </title>
<querytime> 25th February 2011 04:00:00 +0000 </querytime>
<querytweettime> 3857291841983981 </querytweettime>
<querynewesttweet> 3857291841993981 </querynewesttweet>
</top>

10.3 Sample Submission (Adhoc Task)

```
MB01 3857291841983981 1.999 myRun
```

10.4 Sample Submission (Filtering Task)

```
MB01 3857291841983981 1.999 no myRun
```

11. ACKNOWLEDGMENTS

We would like to acknowledge Siena College for the Summer Scholars funding provided to support this research.

12. 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]Miller, G. 1995 WordNet: A Lexical Database for English. *Communications of the ACM* Vol. 38, No. 11: 39-41.

[8]http://trec.nist.gov/pubs/trec11/papers/OVER.FILTERING.pdf

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

[10] Twitter.com