Realtime Ad Hoc Search in Twitter: Know-Center at TREC Microblog Track 2011

Christopher Horn, Oliver Pimas, Michael Granitzer, and Elisabeth Lex

Know-Center Graz, Inffeldgasse 21a, 8010 Graz, Austria
`{chorn,opimas,mgrani,elex}@know-center.at`

Abstract. In this paper, we outline our experiments carried out at the TREC Microblog Track 2011. Our system is based on a plain text index extracted from Tweets crawled from twitter.com. This index has been used to retrieve candidate Tweets for the given topics. The resulting Tweets were post processed and then analyzed using three different approaches: (i) a burst detection approach, (ii) a hashtag analysis, and (iii) a retweet analysis. Our experiments consisted of four runs : firstly, a combination of the Lucene ranking with the burst detection, secondly, a combination of the Lucene ranking, the burst detection, and the hashtag analysis, thirdly, a combination of the Lucene ranking, the burst detection, the hashtag analysis, and the retweet analysis, and fourthly, again a combination of the Lucene ranking with the burst detection but in this case with more sophisticated query language and post processing. We achieved the best MAP values overall with the fourth run.

1 Introduction

Over the past years, the microblogging platform Twitter has gained momentum since people use this platform to share information about current events and about their daily life. In June 2011, Twitter released the information that now 200 million Tweets are sent over Twitter per day[1]. Consequently, it is challenging to find relevant Tweets in this huge amount of data.

Typically, finding relevant Tweets is considered as ad hoc search where users formulate their information needs by means of a query. However, this does not address temporal aspects - for instance, how novel a Tweet is or how novel the information provided in a Tweet is, respectively.

In the TREC Microblog Track 2011, the goal is to rank candidate Tweets not only by topical relevance to a query but also by temporal aspects. More specifically, most recent relevant Tweets should be ranked higher[2].

We propose to address this with three different approaches: (i) a burst detection algorithm, (ii) a hashtag analysis, and (iii) a retweet analysis.

The rest of this paper is structured as follows: Section 3 outlines the Tweet collection, Section 4 describes the three approaches in detail, Section 5 outlines

[1] `http://blog.twitter.com/2011/06/200-million-Tweets-per-day.html`
[2] `https://sites.google.com/site/microblogtrack/2011-guidelines`

the results we achieved for the three approaches, and Section 6 concludes our work and provides an outlook on future research.

2 Task

The goal of the TREC Microblog Track 2011 is to perform realtime adhoc search in Twitter. Firstly, an ad hoc search task has to be performed whereas user information needs are expressed by a set of queries at a specific timestamp. This search task should result in a list of Tweets that are thematically relevant to a query. The real time aspect is addressed so that not only relevant but the most recent Tweets to a query should be ranked highest. As a consequence, to each query, a list of relevant Tweets should be returned that are ordered from most recent to oldest.

3 Collection

The data set for the TREC Microblog Track 2011 consists of the Tweets2011 corpus. The Tweets2011 corpus contains 16 million Tweets from a time period of two weeks (24th January 2011 until 8th February, inclusive). In this time period, two major incidents have been taken place, namely the Egyptian Revolution and the US Super Bowl XLV. The Tweets2011 corpus contains a variety of Tweets: high quality and spam Tweets, replies and retweets. Therefore, it reflects a real sample of the Twittersphere.

In the corpus, each day is represented by a block of about 10,000 chronologically ordered Tweets compressed using gzip. The Tweets are available in both JSON format as well as in HTML format.

We directly downloaded the corpus from Twitter using the HTML crawler from the twitter-corpus-tool[3] provided by the challenge organizers. Therefore, in our approach, we used the Tweets in HTML format.

4 Approach

In our approach, we first indexed the 16 million Tweets using the twitter-corpus-tool[4] provided by the challenge organizers. This resulted in a search index of size 4.1 GB.

Then, we searched the index by the 50 topics provided by the challenge organizers. The 50 topics all exhibit a timestamp; therefore each query represents a specific information need at a specific time.

As search framework, we used Apache Lucene[5], an open source search engine that is able to index several gigabytes of documents at reasonable time. Also,

[3] `https://github.com/lintool/twitter-corpus-tools`
[4] `https://github.com/lintool/twitter-corpus-tools`
[5] `http://lucene.apache.org`

Lucene enables to carry out complex searches quickly on documents in the search index. The results from such a search are already ranked by relevance.

For the TREC Microblog Track 2011, we ranked the search results not only by relevance but also incorporated a burst score, which ranks Tweets within a burst window higher. This covers the temporal aspect of the ranking.

To retrieve relevant Tweets, we applied several search strategies which are described in more detail in the next section.

4.1 Search Strategies

If the topic contained a year, e.g. 2022, we extracted the year using regular expressions and combined the year with the rest of the query terms with AND. For example, the topic is: 2022 FIFA soccer. The according search query is:

(2022 AND (FIFA OR soccer)) AND Timestamp.

The rationale behind that is that typically a year specification corresponds to an important event as for example the 2022 FIFA World Cup.

If the topic consisted of more than two query terms, our restriction was that at least two query terms had to occur. For example, the topic is: NIST computer security. The according search query is:

(NIST AND (computer OR security)) AND Timestamp

However, if with this search query, no or only a few results were retrieved, we repeated the search combining the query terms using OR:

(NIST OR (computer OR security)) AND Timestamp

The resulting Tweets were then subject to several post-processing steps. These are described in the next section.

4.2 Tweet Post-Processing

Since the collection contains not only Tweets in English but also Tweets in other languages as well, we implemented a custom language guesser to filter out Tweets written in other languages. As stated by the challenge organizers, only English Tweets are relevant. Standard language guessers did not work well on the Tweets which is due to the nature of Tweets: they are only 140 characters long and they typically feature a very domain specific language.

Our custom language guesser first removes the links from the Tweets since most links contain English terms which would consequently confuse the language guesser. Then, the language guesser converts the Tweet text to bytes and for each byte, it checks whether the character is an ASCII character or not. Then, the ratio between the number of ASCII characters and total characters is computed. If this ratio is below 0.5 which means that more than the half of characters are non ASCII characters, the Tweet is removed from the search results. Additionally, we derived a set of Spanish, French, and German stop words in order to filter out non English Tweets.

We also filtered out Tweets that contained emoticons, double exclamation marks, and double question marks since these character combinations are clearly an indicator for the Tweet being opinionated [4] and therefore of lower quality [5].

Also, we removed all Tweets from the search results that contained the character @. These are typically replies to other Twitter users. As stated by the challenge organizers, replies are considered as being not relevant to a query. Besides, we computed the Levenshtein distance in order to filter out Tweets that simply repeat information that has already been posted.

After the Tweet post-processing, we applied three different strategies to retrieve not only relevant but also the most interesting Tweets. These three strategies are described in the next sections.

4.3 Burst Detection Algorithm

Content published over Twitter can be regarded as continuous stream of Tweets on arbitrary topics or comments in real time. Due to Twitter's highly dynamic nature, many topics arise, grow in intensity for a certain amount of time and eventually fade away. The identification of a new topic in such a stream of Tweets can be useful to determine the importance of a Tweet. The assumption is that a Tweet is more important if it is the first to discuss a highly influential topic like the Egyptian Revolution, than another Tweet that simply retweets and redistributes the same information. Finding such influential Tweets can be addressed using techniques from the field of burst detection. When determining the appearance of a topic in a stream of documents, this is referred to as *"burst of activity"* [3].

In our approach, we implemented the burst detection by dividing the covered time range of a certain topic into different burst windows. Then, we counted how many Tweets each window contains. If (i) the count exceeds a given threshold, and if (ii) compared to the previous window, the count increases by a certain amount, the particular window is considered to be a burst. All Tweets within that burst window are regarded to be more relevant than Tweets outside this window, hence we ranked them higher. Also, we considered the case where a burst falls in between two burst windows. Therefore, we overlapped the burst windows by a certain amount of time.

Table 4.3 shows the parameters we used for our burst detection approach.

MB009, Title: Toyota Recall

Toyota announced to recall 1.7 million cars due to various issues. Reuters reported about the recall on January 26 2011 at 12:30 CET[6]

Our system detected a burst at the same time:

[6] http://www.reuters.com/article/2011/01/26/us-toyota-recall-idUSTRE70P2EC20110126

Parameter	Value	Description
Window size	3 hours	Size of the burst windows (in hours)
Overlapping time	1 hour	Amount in hours by which windows are overlapping
Burst threshold	4	Necessary amount of Tweets within that window for a window to be considere as a burst.
Minimum increase	2	Necessary increase compared to the previous window

Table 1. Parameters used for Burst Detection

```
Burst with 26 Tweets detected between
   Wed Jan 26 08:13:49 CET 2011 and Wed Jan 26 12:13:49 CET 2011
Burst with 25 Tweets detected between
   Wed Jan 26 17:13:49 CET 2011 and Wed Jan 26 21:13:49 CET 2011
```

MB001, Title: BBC World Service staff cuts

On January 26, 2011, BBC announced to cut 650 jobs in the World Service department[7]. Our system detected a burst with 19 Tweets on January 26 2011 regarding this topic:

```
Burst with 19 Tweets detected between
  Wed Jan 26 09:01 CET 2011 and Wed Jan 26 13:01 CET 2011
Burst with 12 Tweets detected between
  Wed Jan 26 18:01 CET 2011 and Wed Jan 26 22:01 CET 2011
```

Topic MB020, Query: Taco Bell filling lawsuit

On January 24, 2011, a lawsuit against Taco Bell was filed, asking to explain how much actual meat is contained in their meat. Amongst other media, the Chicago Tribune reported this issue on Jan 25 2011, 01:12 CET[8], which corresponds to the output of our burst detection system:

```
Burst with 9 Tweets detected between
  Mon Jan 24 17:20:35 CET 2011 and Mon Jan 24 21:20:35 CET 2011
Burst with 13 Tweets detected between
  Tue Jan 25 05:20:35 CET 2011 and Tue Jan 25 09:20:35 CET 2011
Burst with 43 Tweets detected between
  Tue Jan 25 20:20:35 CET 2011 and Wed Jan 26 00:20:35 CET 2011
```

[7] http://www.bbc.co.uk/news/entertainment-arts-12283356

[8] http://www.chicagotribune.com/news/opinion/ct-talk-taco-bell-0125-20110124,0,4971444.story

Topic MB021, Title: Emanuel residency court rulings

At this given time, the breaking news that Emanuel Rahm is not allowed to candidate as major for Chicago was spread[9]. Our system detected 48 Tweets within a burst window, which is a vast amount of Tweets.

```
Burst with 48 Tweets detected between
Mon Jan 24 18:44:38 CET 2011 and Mon Jan 24 22:44:38 CET 2011
```

These examples show that burst detection can be used to find trending topics in a given data set.

4.4 Hashtag Analysis

Hashtags serve as indicators for a Tweet's meaning, intended audience, or topic [2]. In the past, hashtags have even be used to initiate specific events as for example the recent protests against Wallstreet[10]. In this work, we derived the most prominent hashtag for a topic. The hypothesis is that if a Tweet contains an important hashtag, the Tweet is more relevant.

For the hashtag analysis, we created a hashtag stop word list consisting of the hashtags #ad, #ff, #nowlistening, #sales, #wtf, #nowplaying, #np, #followme, #lastfm, #itunes, since they are often used and they do not contain any relevant semantic information. Then, we used the most often used hashtag as indicator for the relevance of a Tweet; more specifically, if a a Tweet contains the most prominent hashtag, it should be ranked higher.

To finally get a score from the hashtag analysis, we assigned every Tweet that contained the most prominent hashtag a score of 1 and 0 otherwise.

4.5 Retweet Analysis

In Twitter, retweets are used to repeat a piece of information. Therefore, the amount of retweets can be exploited to judge the importance and influence of both a Tweet and a Twitter user [1]. Since we used the HTML version of the Tweets, the retweet count per Tweet was not available; therefore, we had to calculate it on our own. To calculate the number of retweets per Tweet, we used the textual structure of retweets on twitter.

For a certain Tweet t, we searched the whole corpus for the matching expression $e =$ "@ RTauthor-name". This resulted in a collection of all retweets R from that author within our corpus. In order to reduce this collection to only the retweets of t, the trailing text after the matched expression e must match the original Tweet t. However, since Tweets are limited to 140 characters and expression e already takes up five or more characters (assuming that the name exists of one or more characters), there is a chance that a retweet r will only match the first n characters of the original Tweet t. So we defined r a retweet of

[9] http://www.suntimes.com/3469419-417/ballot-booted-court-emanuel-rahm.html

[10] http://tribune.com.pk/story/276605/occupy-wall-street-from-a-single-hashtag-a-protest-circled-the-world/

t if r contains e and all trailing characters of e in r match the first n characters of t, n being the number of characters following the expression e in r. In other words, t must not be equal to but contain the trailing text of e in r.

This method resulted in a list of all Tweets as well as the number of times they were retweeted. However, for a hard real-time task, one can only use the knowledge present at time. So for a specific query, only retweets that existed at the time the query was submitted really suffice this requirement.

In our work, we used the whole corpus to calculate the retweets, which consequently represents future evidence. The reason for using the whole corpus was the runtime-complexity as well as memory requirements of doing the calculation for each and every query.

To finally get a score from our retweets analysis, we assigned every Tweet a score between 0 and 1 (i.e. the retweetScore), relatively to the number it was retweeted.

4.6 Runs

In our approach, we performed four different runs: (i) Run 1, (ii) Run 2, (iii) Run 3, and (iv) Run 4. These runs are described in detail in the next sections.

Run 1 For Run 1, we first used Lucene to query and rank the Tweets. After that, we performed the burst detection, identifying the time windows (interval: 3 hours) when an unusual amount of Tweets occurred. Based on our assumption that something extraordinary happened within that time, those Tweets have been ranked higher by a burst detection score (BurstScore).

$$BurstDetectionScore = LuceneScore * 0.51 + BurstScore * 0.49; \tag{1}$$

where the $BateScore$ is 1 if a Tweet is in the burst window and 0 otherwise.

Our final score combines both Lucene score and the burst detection score. If no burst was detected, only the Lucene score was considered.

Run 2 For Run 2, we again first used Lucene to query and rank the Tweets. After that, we performed the burst detection, identifying the time windows (interval: 3 hours) when an unusual amount of Tweets occurred.

As second ranking factor, we counted the number of retweets (retweet score) of all Twitter user over the whole corpus. Note that we did not restrict this computation to retweets up the given query time. As third ranking factor, we computed the most often used hashtag per topic and ranked Tweets higher that contained the most often used hashtag (hashtag score). Note that this computation has been done only on Tweets posted before the query timestamp (no future evidence). Our final score combines therefore a Lucene score, the burst detection score, the retweet score, and the hashtag score. If no burst was detected, only the other three score have been considered.

Our final score combines therefore a Lucene score, the burst detection score, and the retweet score.

$$Score = LuceneScore * 0.8 + RetweetScore * 0.2; \tag{2}$$

Run 3 For Run 3, we again first used Lucene to query and rank the Tweets. After that, we performed the burst detection, identifying the time windows (interval: 3 hours) when an unusual amount of Tweets occurred. As second ranking factor, we computed the most often used hashtag per topic and ranked Tweets higher that contained the most often used hashtag (hashtag score). Note that this computation has been done only on Tweets posted before the query timestamp (no future evidence).

Our final score combines therefore a Lucene score, the burst detection score, and the hashtag score:

$$Score = LuceneScore + HashtagScore * 0.005; \tag{3}$$

If no burst was detected, only the other two scores have been considered.

Run 4 For Run 4, we again first used Lucene to query and rank the Tweets. After that, we performed the burst detection, identifying the time windows (interval: 3 hours) when an unusual amount of Tweets occurred.

Our final score combines therefore a Lucene score and the burst detection score. If no burst was detected, only the lucene score have been considered.

The difference to Run 1 is the use of the Levenshtein distance in order to filter out too similar Tweets in the final ranking. The Levenshtein distance has been computed between the current Tweet and the last Tweet whereas the current Tweet and the last Tweet must differ by at least 50% of their characters. Also, in this run, we used more sophisticated stop word lists (slang, emoticons, hashtags...).

5 Results

The overall MAP results achieved for all four runs are presented in Table 5: The best results in terms of MAP over all topics were achieved with Run 4. So

Run	MAP
1	0.1797
2	0.1916
3	0.1916
4	0.1930

Table 2. MAP results for all four runs.

apparently, exploiting only the Lucene relevance ranking in combination with our burst detection approach gives the best results. The second best results were achieved with Run 2 and Run 3. In the following, we concentrate on Run 3 since in Run 2, future evidence has been used. More specifically, we compare Run 3 with Run 4 in respect to a subset of the given topics.

Table 5 details results achieved in Run 3 and Run 4 for a selection of topics:

Topic	Query	P30 Score Run 4	P30 Score Run 3	Burst found
MB007	Pakistan diplomat arrest murder	0.8667	0.8667	yes
MB008	phone hacking British politicians	0.5333	0.5333	yes
MB036	Moscow airport bombing	0.6000	0.6000	yes
MB041	Obama birth certificate	0.3333	0.3333	yes
MB003	Haiti Aristide return	0.6667	0.7000	no

Table 3. Results for Run 3 and Run 4 for selected topics (results taken from allrel).

Our approach performed very well for some queries: for instance, for topic MB007, we achieved a precision on the 30 highest ranked topics (P@30) of 86% with Run 3. The median P@30 over all submissions is 60%. As can be derived from the table, including the hashtag score did not improve the results for this topic. This can be interpreted that the relevant Tweets did not contain the most prominent hashtag or that no hashtag existed for the relevant Tweets.

In case of topic MB003, no burst was detected but the hashtag analysis improved the P@30 from 0.6667 (Run 4) to 0.7000 (Run 3.) . The hashtag in this case was #haiti, a clearly topically related term.

We investigated the resulting MAP values for our four runs. For topic MB021, we achieved a high MAP value of 0.2794. For this topic, we detected 7 bursts. In this case, our burst detection approach was therefore successful.

For topic MB048, we achieved a low MAP of 0.0040. We investigated the results of our approach and we found that for this topic, a large amount of bursts (18 bursts) were detected by our system. We checked the Tweets contained in the bursts and we found that the Tweets tackled the Egyptian revolution but not the specific topic Egyptian evacuation. Clearly, in this case, with the burst detection, we identified the general topic of the Egyptian revolution, but not the subtopic of the evacuations. Therefore, our approach was not successful in that specific case. The challenge here clearly is how to identify the most relevant bursts.

6 Conclusions

In conclusion, we presented our approach towards the TREC Microblog Track 2011. We proposed a retrieval technique in combination with three approaches: a burst detection, a retweet analysis, and a hashtag analysis. Our experiments revealed that for certain topics, we achieve high MAP values using the burst detection. If too many bursts are detected, it is challenging to identify which bursts are more relevant than others. In our setting, we treated all bursts equally. For future work, we aim to distinguish relevant bursts from non relevant bursts. Also, for certain topics, incorporating the hashtag score was beneficial: frequently used hashtags indicate the semantic relation to a topic. However, boosting Tweets with the wrong hashtags negatively influences the ranking. So, the identification of hashtags relevant for topics is crucial.

7 Acknowledgements

The Know-Center GmbH Graz is funded within the Austrian COMET Program - Competence Centers for Excellent Technologies - under the auspices of the Austrian Ministry of Transport, Innovation and Technology, the Austrian Ministry of Economics and Labor and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

References

1. D. Boyd, S. Golder, and G. Lotan. Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. *Hawaii International Conference on System Sciences*, 0:1–10, 2010.
2. M. Efron. Hashtag retrieval in a microblogging environment. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 787–788, New York, NY, USA, 2010. ACM.
3. J. Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 91–101, New York, NY, USA, 2002. ACM.
4. E. Lex, M. Granitzer, M. Muhr, and A. Juffinger. Stylometric features for emotion level classification in news related blogs. In *Proceedings of the 9th RIAO Conference (RIAO 2010)*, 2010.
5. E. Lex, A. Juffinger, and M. Granitzer. Objectivity classification in online media. In *HT '10: Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 293–294, New York, NY, USA, 2010. ACM.