University of Glasgow (uog_tw) at TREC Microblog 2012

Jesus A. Rodriguez Perez, Andrew J. McMinn, and Joemon M. Jose
{j.rodriguez-perez.1, a.mcminn.1}@research.gla.ac.uk, joemon.jose@glasgow.ac.uk

School of Computing Science
University of Glasgow
Glasgow, UK
G12 8QQ

ABSTRACT

In TREC 2012, we participated in the search and filtering tasks of the Microblog Track. The Microblog track is in its second year, and whilst the real-time ad-hoc search task remained very similar to 2011, a new real-time filtering task was introduced. For both tasks, we made use of temporal evidence and a burst detection algorithm to re-rank documents created during bursty periods. We also made extensive use of query expansion, which yielded good results in the ad-hoc task, but mixed results in the filtering task. Many of our runs were well above the TREC Median, and the methods we propose show promise for application to Twitter.

1. INTRODUCTION

Microblogging has grown in popularity in recent years, and is gradually transforming the way we communicate with each other, share information, and find out about events. Twitter is a particularly interesting service because it encourages users to discuss events in real-time, and often includes first hand reports of an event as it is developing. This allows for a unique insight into events from the subjective opinions of those directly involved to the discussion between others following the event through social and traditional media.

Ad-hoc retrieval is one of the the most commonly researched tasks in Information Retrieval, where the goal is to return documents that are relevant to an immediate information need, expressed as a query. Tweets are limited to 140 characters in length, and over 340 million tweets are posted every day. Tweets are generally of a low quality as they contain bad grammar or spelling mistakes, and slang is often used to overcome their restricted length.

The combination of these factors makes searching in Twitter extremely difficult, causing traditional ranking models, such as TF-IDF [1], to experience difficulty in retrieving the most relevant documents.

For our experiments in the real-time ad-hoc and real-time filtering tasks, we employed various different techniques. Our main focus was on temporal features and query expansion techniques in the challenging context of tweet retrieval.

2. TWEETS11 CORPUS

The Tweets11 collection was once again used for the 2012 track, however, in the interest of fairness, the collection was re-downloaded from Twitter shortly before the May 7 cutoff time. This was because Tweets often become inaccessible over time due to deletions and changes in their public visibility. This would have disadvantaged newcomers who did not already have a copy of the corpus as they would not have had access to the full collection.

The corpus spans a two-week period from the 24th of January to the 8th of February 2011. It contains approximately 16 million Tweets across all languages, and is designed to be a representative sample of Twitter, containing both useful and spam tweets.

2.1 Spam and Language Filtering

We performed basic filtering to remove spam and non-English Tweets from the Tweets11 corpus. Tweets which contained more than 3 hashtags, 3 mentions, or 2 URLs were classified as spam and removed. This decision was based upon statistical observations of the corpus, and thorough careful examination of the most common types of spam on Twitter.

Non-English tweets were removed based upon two factors: (1) the author's primary language, and (2) the language of the tweet. The author's language was taken directly from their profile data, which is included in the JSON of every tweet. However, we found that simply relying on the language specified by the user was very unreliable, and that a large percentage of users did not write tweets in the language specified by their profile. In particular, a disproportionately large number of users identified their language as English, but wrote tweets in another language. The language used in individual tweets was identified using a Java language-detection library[1] which uses naive Bayesian filtering and claims over 99% precision for 53 languages.

After filtering, around 4 million of the original 16 million Tweets remained. This gave us a clean base from which to run our experiments, and greatly reduced the time and space required to index the collection.

3. BURST DETECTION ALGORITHM

We believe that ad-hoc retrieval performance in microblog streams can be improved by taking the temporal dimension of the documents into consideration. In order to do this, we need two components. Firstly, a burst detection algorithm from which we can extract temporal features during bursts. Secondly, a retrieval model which incorporates the temporal features to weight terms and rank relevant documents.

3.1 Burst detection

[1] http://code.google.com/p/language-detection/

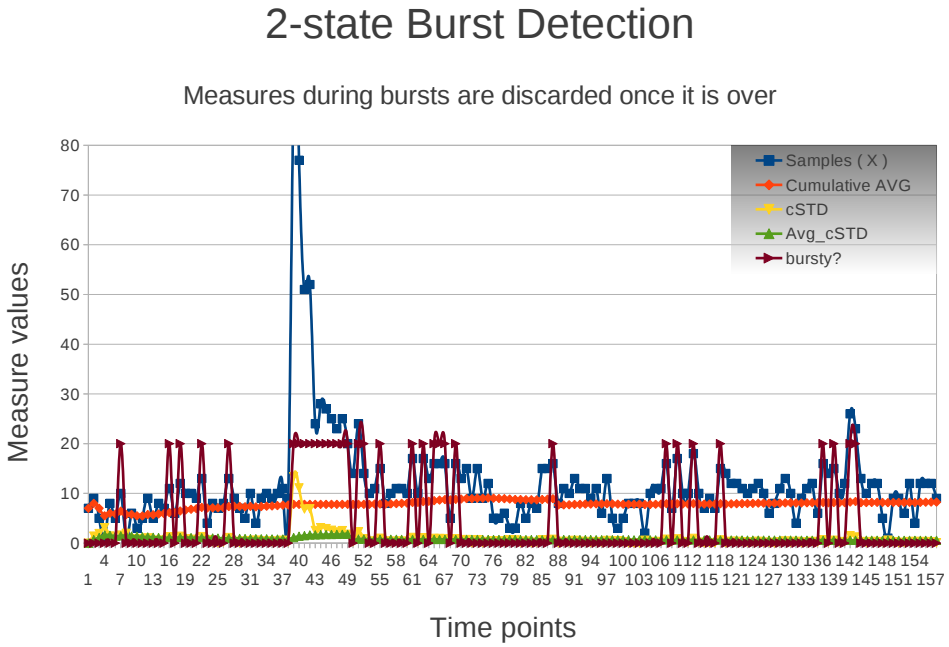

Figure 1: Example of burst detection using real data for the term "Airport"

Our approach examines the timeline of terms searching for sudden positive changes in the frequency, which can represent a burst. To identify a burst, we produce two signals: one which carries the historical rate of change of the timeline, and another which reflects to current rate of change. To this end, we propose the use of the relationship between two cumulative statistical measures, namely cumulative Standard Deviation ($cSTD$), with the current rate of change and average of the cumulative Standard Deviation (Avg_cSTD) with the averaged historical rate of change. Computing these signals for each hour long time windows, we can evaluate whether a term has bursted at a particular time window. All the measures utilised are formalised below:

$$cAvg_{t,i} = \frac{X_{t,i} + (i-1)(cAvg_{t,i-1})}{i} \quad (1)$$

$$cSTD_{t,i} = \frac{cSTD_{t,i-1} + (X_{t,i} - Avg_{t,i-1})(X_{t,i} - cSTD_{t,i})}{i} \quad (2)$$

$$Avg\_cSTD_{t,i} = Avg(cSTD_{t,i})_{t,i} \quad (3)$$

where the frequency X of term t is defined as the times t is mentioned during a time window i. Moreover, we compute the arithmetic cumulative mean ($cAvg$) as the frequency for term t up to time window i. Likewise, we compute $cSTD$ for a term t for the given time window i. Finally Avg_cSTD is computed for the same window i and term t.

$cSTD$ represents the cumulative standard deviation of the frequency for a given term t at a give time window i within the time series. It is computed using Formula 2. Notice that this formula depends on the cumulative arithmetic mean of the term frequency up to time window i (Defined in Formula 1), as well as the $cSTD$ value from the previous time window.

As Avg_cSTD gives us a stable measure of the historical rate of change, we consider that a term t is bursty at time window i when the $cSTD$ signal is larger than Avg_cSTD (i.e. the current rate of change is higher than that previously observed). Note that whilst equations 1 and 2 have been used by [3], we provide Avg_cSTD allowing our system to determine whether a term is bursting or not.

The main issue with these statistical measures is that smaller bursts can be smothered if proceeded by a larger burst. The measures may become numb as the cumulative arithmetic mean takes into account the sudden change in frequency. To address this problem, we borrow an idea from [2], by which bursts are modelled as states of a state machine. Our model contains two states, *idle* and *bursting*. The transition from *idle* to *bursting* is done when $cSTD$ is higher than Avg_cSTD. Likewise, the transition from *bursting* to *idle* occurs when the value for $cSTD$ drops below Avg_cSTD.

When transitioning from *idle* to *bursting*, the values of all measures are stored in the *idle* state. These values are later restored when the *idle* state is re-established, discarding the values obtained during bursts from future computations, which results in better sensitivity to bursts. This approach is formalised in Algorithm 2.

Since no previous information for a term is available when it appears first time, the signals require between 3 and 5 samples to stabilize. Consequently during this range of time, bursts are not considered as they are very likely to be a false positives. To help minimize this loss, a global average frequency across all terms is maintained throughout all time windows, and it is used as a preinitialization value for a new incoming term. In other words, if the $cSTD$ signal is larger than Avg_cSTD when using the sampled frequency for the new term, and the global average, then a burst may be considered at start.

Figure 1 illustrates how the burst detection algorithm performs during a real scenario. This figure includes the above

Figure 2: Burst Detection algorithm pseudo code

```
input: term t
input: time window i
foreach time window i do
    foreach term t in window i do
        cAvg(t,i) = cumulative avg frequency up to i for term t
        cSTD(t,i) = cumulative standard deviation of frequency for t up to i
        Avg_cSTD(t,w) = cumulative avg of cSTD(t,i) for t up to i
        if cSTD > Avg_cSTD then
            current_state = "bursty"
            if old_state is not "bursty" then
                store current values
            end
        else
            if old_state is "bursty" then
                restore old_state values
            end
        end
    end
end
```

mentioned measures, as well as the decision by our algorithm denoted by *bursty?*. A term is considered "bursty" at a particular time window, when the *bursty?* measure, is higher than 0. We can observe how it correlates quite closely with the bursts we may evaluate manually with respect to the frequency (*"Samples (X)"* measure in Figure 1).

One of this method's main advantages is its low storage and computational complexity. The measures are cumulative, and as such, we only require the most recent time window and a reference window, eliminating the need for historical frequency information during the burst computation. Furthermore, all terms are dealt with independently, simplifying the computational complexity greatly. This enables the burst detection algorithm to scale well, and allows it to work in real-time on modest hardware.

During a time window, when a burst is identified we store three pieces of information: The start time of the burst, the end time of the burst, and its magnitude, which is defined as:

$$mag_{t,i} = Avg\_cSTD_{t,i} - cSTD_{t,i} \quad (4)$$

Finally, this information is kept in an inverted index where the terms are the keys and the bursts information is stored as values, and later used by the retrieval models.

4. REAL-TIME AD-HOC TASK

Real-time ad-hoc tasks are based on the scenario where a user wants the latest information with respect to a textual query. Therefore the participating systems in this task should return the best documents to satisfy the user's information need, ordered from newest to oldest.

After preprocessing all tweets, we indexed the collection using the Terrier IR [4] platform and its double pass indexing option. The information kept in the index was comprised of the text and the tweet ids, which are later used for the retrieval. We chose to use Terrier IR as it provides a Java API and a variety of retrieval models which we could use out of the box.

When choosing our baseline, we experimented with multiple traditional retrieval models including *tf*idf*, *idf*, *bm25* and *Hiemstra's Language Model*, etc. We found that IDF outperformed all other models due to its independence with respect to document specific metrics. This motivated the use of *idf* as our baseline, since models relying on document specific metrics, in the tweet retrieval context, performed much worse in terms of precision, recall and MAP.

4.1 RUN1: IDF Only

This run was used as our baseline, and uses the IDF retrieval model over the filtered collection, with the resulting list of tweets ordered according to recency (Most recent document first).

4.2 RUN2: IDF and Query Expansion by PRF

This run builds upon the first run, and extends the search providing a query expansion approach based on pseudo relevance feedback (PRF). PRF assumes the first N documents are relevant to the information need posed by the user by means of the query. We empirically set N to be the first 3 tweets as it becomes counter productive to consider more documents in this context, resulting in worse performance. When selecting a term for query expansion, we adopt a very conservative posture, considering only terms which appear at least in two of the three tweets.

4.3 RUN3: IDF and Burst Detection

This run utilises the temporal information provided by the burst detection algorithm described previously, to reorder results according to their temporal relevance. To accomplish this, retrieval performed in two stages. Firstly, a query is issued and a ranked list of documents is generated for each of the topics using *idf*. Secondly we analyse the query terms looking for bursts in frequency they might have had during the same time the tweets in the list were published. Each of the time windows in which the documents were published will then have a score which reflects the burstiness of the query terms at that particular time. We compute the score using this equation:

$$Score_i = N_i + \frac{\sum_{w=1}^{\text{total\_terms}} mag_{w,i}}{100} \quad (5)$$

Where w refers to a word, i refers to a time window and mag the magnitude of the burst for a term w during a window i. There are two aspects we wanted to reflect in this equation. First the co-ocurrency of bursty terms is promoted, giving higher scores to documents containing more bursty query terms through N. Secondly, to characterize the temporal impact of the bursts, the equation adds together the magnitudes of the bursts experieced by the query terms during that time window. Finally we have empirically seen that the sum of magnitudes is never higher than 100, thus to use this feature as a second tier in the reranking, we divide the sum of the magnitudes by 100.

Due to having a fixed window size and the time required for detecting a burst, it is possible for an important number of relevant tweets to lie in the temporal vicinity of a detected burst. We attempt to minimize this issue by assigning scores to tweets located near the bursty windows, using a linear interpolation approach. Specifically, we interpolate a score for the tweet contained in the temporal vicinity using the range [$\text{mag}_{w,i}$...0]

5. REAL-TIME FILTERING PILOT TASK

The aim in the real-time filtering tasks is to decided if a subsequently posted tweet is relevant to a query submitted

Table 2: This table shows the results achieved for each of the runs performed as part of the filtering task.

Run	Submitted?	***Scaled Utility***	***F (beta=0.5)***	***Set-Precision***	***Set-Recall***
TREC Median	yes	0.2076	0.1491	0.1766	0.3343
FRUN1	yes	0.2072	0.2297	0.2657	0.4238
FRUN2	yes	0.0617	0.1017	0.1099	**0.4928**
FRUN3	yes	**0.2590**	**0.2835**	**0.3414**	0.4440

Table 1: This table shows the results achieved for the real-time ad-hoc task.

Run	Submitted?	***MAP***	***P@30***	***R-prec***
TREC Median	yes	—	0.1509	**0.1905**
RUN1	yes	0.0978	0.1192	0.1270
RUN2	yes	0.1089	0.1333	0.1387
RUN3	yes	**0.1297**	**0.1582**	0.1567
IDF+IBF	no	0.1087	0.3226	0.1570

at particular point in time. In this task, the assumption is that the user has already seen a tweet, or tweets, related to a topic they are interested in. The user now wishes to find new relevant tweets to keep them informed of any developments.

For these runs, we made use of the ability to receive immediate relevance feedback for the users.

5.1 FRUN1: IBF

Events are highly temporal, therefore we assume that utilizing temporally sensitive approaches to retrieval, can improve the retrieval performance for event based queries. *Inverse Burst Frequency* (IBF) works under the assumption that temporal data can be used to compute the discriminative power of a term, analogously to *idf*. To this end, IBF uses data provided by the burst detection algorithm following the structure set by *idf*, as formalized in equation (6).

$$IBF_{t,B} = \frac{|B|}{|b \in B : t \in b| + 1} \tag{6}$$

where t refers to the term, B the set of all bursts recorded by all the query terms, b represents a burst and $|B|$ is the total number of bursts identified in the collection. Alternatively B can include the bursts produced by all terms, but we achieved best results when we used query terms only.

Similar to *idf*, the more often a term is identified as bursty, the lower the score assigned to the term. Therefore, terms that burst infrequently are assumed to be more discriminative of the documents we want to retrieve.

5.2 FRUN2: IBF & Query Expansion by RF

This run builds upon last run, by using the relevance feedback we were allowed to use in this particular task. When the system assessed a document as being relevant to the query at hand, the system was allowed to instantly know if its decision was correct, providing an incredibly fast and quite unrealistic relevance feedback.

5.3 FRUN3: Adaptive Query Expansion

As each new Tweet is received, a score is calculated for the Tweet d against query q. If $score(d, q)$ is greater than 0.5, the Tweet is displayed to the user, who is then able to judge it as relevant or non-relevant. Query expansion beings once 2 documents have been marked as relevant by the user, and is revised after every subsequent judgment.

5.3.1 Query Expansion

We take the most frequently occurring terms which appear in at least $\frac{1}{3}$rd of the documents marked as relevant by the user, and append them to the query. New query terms are added to the original query until the query reaches a maximum length of 5 terms. For example, the query *"Mexico drug war"* is capable of being expanded by up-to 2 terms, however queries such as *"phone hacking British politicians voicemail Prime Minister Gordon Brown police"* will receive no expansion.

5.3.2 Scoring

Scoring was performed using query length normalized TF. Although term frequency is known to perform poorly in retrieval tasks when applied to Twitter, its performance seems considerably better in filtering tasks.

The original query terms are double weighted when calculating a document's score. This is to reduce the likelihood that the results will be skewed too far from the original query, whilst still allowing for a reasonable difference to be made by the expanded terms.

The score for a document d and query q is given as:

$$score(d, q) = \frac{d \cdot q}{|q|}$$

6. RESULTS AND DISCUSSION

In this section, we present the results obtained for each of our experiments in the Microblog Track. Many of our submitted runs in the filtering task out-performed the TREC Median, and development after the TREC deadline allowed us to further improve our algorithm's performance in the ad-hoc search task.

6.1 Real-time Ad-hoc Task

Table 1 shows the results of our experiments in the real-time ad-hoc retrieval task. This year's standard measurements, as agreed by TREC organizers, are given. Note that we have also included results for an additional run which was performed after improvements to our algorithm after the ad-hoc deadline.

From the results obtained, we can see that only our third run performs better than the TREC Median in terms of precision at 30. In terms of R-precision, all of our submitted ad-hoc runs perform worse than the TREC Media. However, our additional experiments with *IDF+IBF*, which is based on the same principles as runs 2 and 3, achieves twice the precision at 30 compared to the TREC Median – an encouraging result. However, the R-precision performance of our newest approach remains lower than the median, and near identical to our best submitted run. We believe this may be due to our system being able to target highly relevant documents, which is easily reflected in terms of precision, but does not have an effect in terms of recall. Thus R-precision

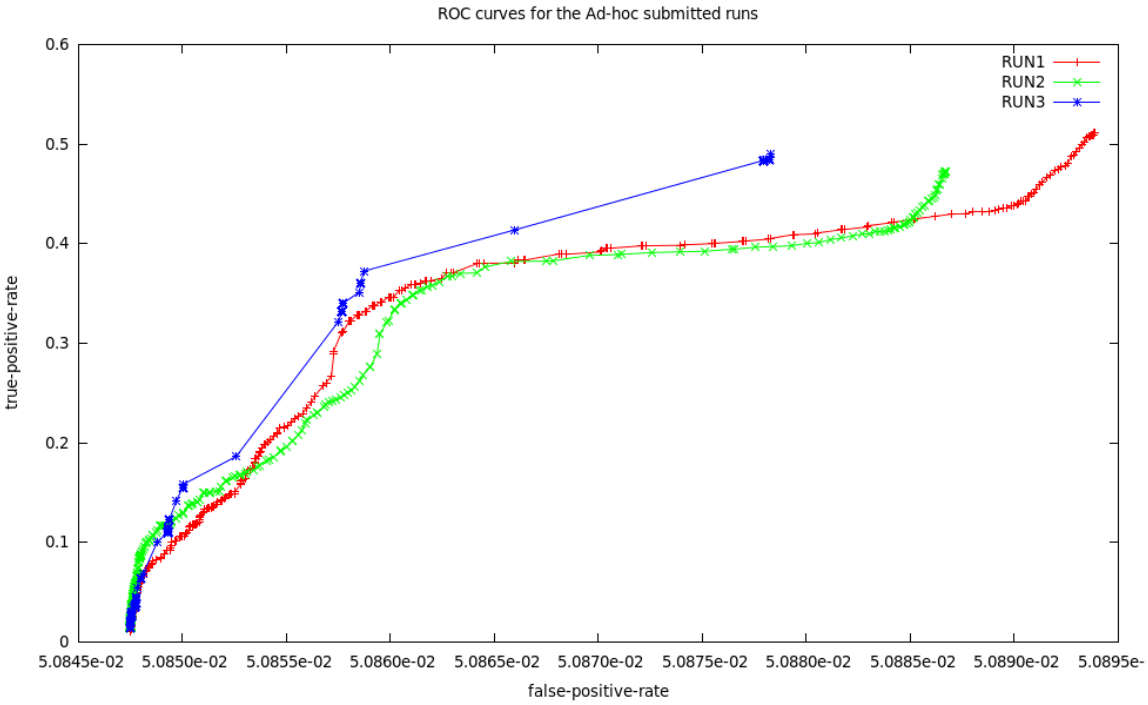

Figure 3: Roc curves for submitted runs

does not correlate with the improvement in performance experienced by the precision measures in our runs.

Finally, a comparison is made in Figure 3, which shows the plot of ROC curves for each of the submitted runs. Here, we can see how RUN3 gives the best performance. This means that the rate of false positives is lower with respect to true positives when compared to the other two runs. Following the same reasoning, we can see that the second best performing run in terms of the ROC is RUN2, as it is closer to the left than RUN1, leaving RUN 1 in third place.

6.2 Real-time Filtering task

Table 2 shows the results for the filtering task. The addition of more terms as part of query expansion in FRUN2 may have been counter productive, as many terms do not appear to show signs of burstiness. As a result, we believe that IBF is better suited as complementary weighting when combined with more traditional ranking methods. In particularly, we have had success pairing it with IDF weighting.

Our Adaptive Query Expansion, which was used by FRUN3, seems to provide the best results by a significant margin - with its F and Precision measures being almost double that of the TREC Median. The performance of the query-length normalized TF scoring is surprising, especially given that TF has been show to be one of the worst performing weighting schemes for ad-hoc retrieval on Twitter. Further investigation is needed to compare other scoring systems, such as IDF, with the adaptive query expansion method described.

7. CONCLUSION

In this paper, we have presented a burst detection technique and query expansion techniques for use with Twitter. We applied our burst detection technique to both the ad-hoc and filtering tasks which showed significant improvements over the baseline IDF ranking. Further investigation, after the ad-hoc deadline, produced the IBF ranking scheme, which showed further improvements over the baseline.

Experiments with query expansion yielded slight improvements in the ad-hoc search task, and mixed results in the filtering task. The different query expansion algorithms used in the filtering runs produced vastly differing results, suggesting that query expansion can be very beneficial, or very detrimental depending heavily on the method of expansion and the data available. Overall, IBF and our query expansion techniques produced results significantly higher than the TREC Median in the filtering task, and shows that they have great potential for use on Twitter.

8. REFERENCES

[1] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[2] J. Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

[3] M. Naaman, H. Becker, and L. Gravano. Hip and trendy: Characterizing emerging trends on twitter. *Journal of the American Society for Information Science and Technology*, 62(5):902–918, 2011.

[4] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proceedings SIGIR'06 Workshop (OSIR 2006)*, 2006.