Using Stream Features for Instant Document Filtering

Andreas Bauer
Media Informatics Group
University of Regensburg
Regensburg, Germany
andreas.bauer@extern.ur.de

Christian Wolff
Media Informatics Group
University of Regensburg
Regensburg, Germany
christian.wolff@ur.de

Abstract

In this paper, we discuss how event processing technologies can be employed for real-time text stream processing and information filtering in the context of the TREC 2012 microblog task. After introducing basic characteristics of stream and event processing, the technical architecture of our text stream analysis engine is presented. Employing well-known term weighting schemes from document-centric text retrieval for temporally dynamic text streams is discussed next, giving details of the ESPER Event Processing Agents (EPAs) we have implemented for this task. Finally, we describe our experimental setup, give details on the TREC microblog runs as well as the result thereafter with our system including some extensions and give a short interpretation of the evaluation results.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

Experimentation

Keywords

information filtering, event processing, web2.0, text streams, real-time search, tf/idf, okapi, stream features

1. INTRODUCTION

Due to the rapid growth in user-generated digital content, data stream processing has received increasing scholarly attention. Most of this content is textual, thus investigating how to effectively rank real-time text streams is an interesting research question.

In this paper we present an event-based approach to real-time information filtering as well as our results created for the microblog real-time filtering task for TREC 2012. In addition, we discuss improved results, which we have achieved after the TREC 2012 microblog task deadline. We also show how to leverage (complex) event processing engines for continuous term weighting and feature generation.

2. EVENT-BASED INFORMATION FILTERING

The basic idea of splitting and mapping text streams onto semantically distinct event types has already been presented in [1]. In short, the approach here is to feed an incoming tweet into a network of event processing agents that immediately execute the analysis and provide an instant ranking of the tweet. This is possible because modern event processing engines like *Esper*[1], *Tibco* BusinessEvents[2] or *Drools Fusion*[3] support high-speed processing of events.

More recently a streaming version for the big data framework *Hadoop* has been released[4] and *Twitter* has published its real-time streaming system *Storm*[5]. In addition, the S4 distributes streaming platform, originally published by Yahoo, is now an incubator project supported by the Apache Foundation as well.[6]. All these developments show that event processing is still needed and is consider as a viable and elementary part of the efficient processing of large amounts of data. Big Data and event processing are no mutually exclusive concepts but rather complementary, where event processing addresses interesting goals in analysing large amount of data by offering features like sliding windows or pattern matching.

It is quite obvious that streaming and event processing offer major opportunities for analysing text streams in real-time and that the industry is not only focusing on increasing the speed of analysing large amount of data. While the event processing paradigm as proposed by David Luckham[9] originally focussed on business applications, the applicability for information retrieval tasks has been recognized in more recent work[3, p.10][10, p. 42]. We have used *Esper* as our event processing engine of choice, because it is open source, offers good online support and allows for a straightforward

[1] http://www.espertech.com
[2] http://www.tibco.com/products/event-processing/complex-event-processing/businessevents/default.jsp
[3] https://www.jboss.org/drools/drools-fusion.html
[4] http://hadoop.apache.org/docs/r0.15.2/streaming.html
[5] http://engineering.twitter.com/2011/08/storm-is-coming-more-details-and-plans.html
[6] http://incubator.apache.org/s4/

integration of high volume text stream analysis. Some of the advantages of an event processing approach are:

1. *Clear semantics*: Due to the real-time scenario each digital utterance has to be considered in its temporal context. The sheer amount of information constantly generated raises the probability of an information being lost, overlooked or ignored increases as time passes by. The event metaphor is well suited for this type of informational scenario because it stresses the temporal aspect.

2. *Interoperability*: From a design point of view the mapping of text onto event types allows for easy combination of events from different sources. E.g. if Facebook status updates and Tweets are mapped onto the same basic event type like *Token Event*, *Location Event*, or *Sentiment Event*, cross data stream analysis can easily be performed, because the same event processing agents (EPAs) as well as the same event processing network (EPN) can be used.

3. *Technical integration*: Many current *big data* or large-scale analysis systems rely on exploiting the temporal nature of information. Thus, they are designed to work with events that can be analysed in a temporal manner by providing temporal meta-information, e.g. detection time, creation time or expiry time.

3. TECHNICAL ARCHITECTURE

Figure 1 shows the overall technical architecture of the system we have used for our experiments. Mircoblogs are

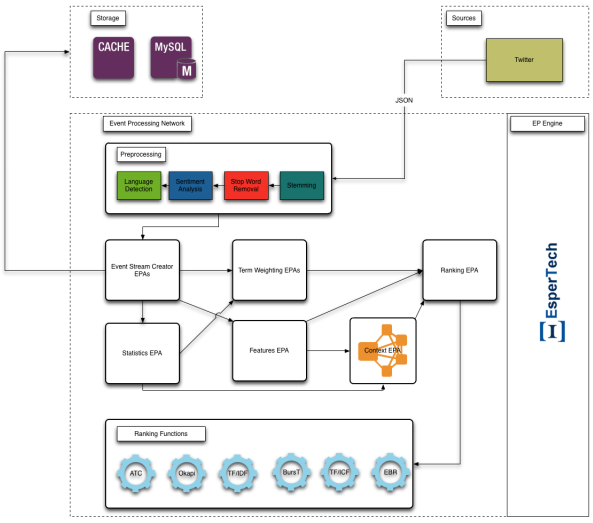

Figure 1: Architecture Overview

imported via a JSON interface and fed into the text preprocessing component. which includes the following steps:

1. Conversion to lower case
2. Stemming with the Porter stemmer
3. Stop word removal[7].
4. Enriching basic tweet events with statistics on upper/lower case characters, character count and stop word count.
5. Language detection

After the conversion of raw Tweets into $TweetEvents$, the latter are fed into the event processing network using the *ESPER* event processing engine. First, each tweet is sent to an event processing agent that splits up the text of the tweet and maps it onto semantically distinct event types. Then these events are processed by several other Event Processing Agents, that calculate stream statistics like average text length, count of distinct tokens, term counts, etc.

Important to notice are the $SearchProfileEPAs$ that were created for every TREC topic. Within these EPAs the filtering was done which was based on different ranking schemes. Important to fulfil the real-time requirement the amount of Tweets that should be examined was decrease by a first filtering step that prevent many Tweets from entering the real filtering stage. This first filtering step has major influence on the recall of the system, because it acts as a gatekeeper to the filtering processes. Hence a strict filter condition increases precision, but lowers recall for instance. For the runs we present in this paper the primary filtering rule is shown in condition 1

$$Filter = \{x | qm \geq 1 \vee stl = 1 \vee (qm > 0 \wedge fm > 1)\} \quad (1)$$

With qm being a term match of a topic term and a term in the Tweet, stl being a search profile with only one search term and fm being a feedback match, i.e. a feedback term was encountered in an incoming Tweet.

4. USING STREAM FEATURES FOR RANKING

The stream features that were described in the previous section are now processed using different ranking algorithms. For the TREC 2012 runs, we were only able to provide two algorithms, one based on *OkapiBM25* and the other a standard Vector Space Model-based approach. For each schema we have provided one run with and without relevance feedback. In later publications we will present the application of stream based features using *BursT*, *TF/ICF* and incremental *TF/IDF*. Furthermore we will present the results of plugging stream based *TF/IDF* values into weighting schemes like *ATC* and *LTU*[5, p. 4]. Here, we would like to find out whether stream based features calculated by applying sliding time windows onto the data streams offer reasonable results.

This is relevant because this approach emphasizes the temporal sensitivity of events, i.e. a streamed approach reflects better the continuous nature of a text stream.

4.1 Components of a Document Ranking Scheme

[16, p. 517f] state that there are three relevant components within a term weighting scheme: *Term frequency* describes what a document is about. The second component is a factor that reflects the distribution of terms in the document collection as a whole while the third factor takes into account the length of a document in order to avoid over-

[7]Stop word list used: `ftp://ftp.cs.cornell.edu/pub/smart/english.stop`

or under representing terms in the weighting scheme (document length normalization). All of these components were taken into account and adjusted for the real-time filtering scenario. In the following subsections we discuss how concepts like term, document, document collection and relationships have to be adjusted for the microblog scenario and its streaming and real-time aspects.

4.1.1 Determining term frequency in text streams

[12, p. 2] have showed that 85% of all Tweets contain terms only once. Hence the traditional term frequency measures that rely on the term count within a document fail, because for almost every token within a tweet the *tf* value would be the same. To overcome this problem we have used stream statistics in order to derive a *term frequency replacement value* that reflects the importance of a term. As mentioned above we use sliding time windows: With these windows we build a windows into the past that reflect the last k seconds, minutes or events of the data stream. We think that this approach reflects best the temporal notion of information streams.

For the runs we submitted to TREC 2011 we have used a sliding time window of 120 seconds to calculate ad-hoc statistics for each event stream. Assuming we could use approx. 4.5 out of 12 million downloaded Tweets distributed equally over 16 days, we get an average event frequency of approx. 300k events per day. Assuming a constant event arrival rate, this means that only 4 events arrive per second.

In the set-up for the TREC runs we have set the event arrival rate to 500 events per second, because otherwise a run would have taken understandably 16 days, if we had kept the original arrival gaps. Extrapolating the time window of 120 seconds, which we used for the TREC runs, this corresponds to a real world window of approx. 4 hours. This value was chosen arbitrarily and will be subject of further investigations. In general, we can assume that the smaller a time window is the better it reflects the most recent changes.

For the post-TREC runs – runs we created after the deadline of TREC – we changed the values to 1200 events per second. The best results, which will be presented in the last section5.1.1, were yielded with a time window of 240 seconds. If we extrapolate this to *real time* this would correspond to a sliding time window of almost one day.

We use the sliding time windows to build dynamic statistics for different events type of the stream: We have built the top-k window for hashtags and tokens event stream. For the presented runs we only used hashtags and tokens for calculating a local term weighting factor, because hashtags are ""derived"" from regular tokens and hence can be used for the boolean matching step that is conducted in order to determine the terms that should be weighted.

There are more special semantics to Twitter like retweets(*rtusername*), mentions (@*username*) and or embedded links, that could be used for filtering the stream, but this is still subject to further investigation.

Listing 1[8] shows how to construct the top-k ranking for retweets. Each *insert into* statement can be considered as a separate EPA. Esper offers the possibility to subscribe to such statements and then Java code can be executed on the events that are being processed by the EPA.

[8] 1 shows how the top-k retweets are generated.

Stream	Weight
Token Stream	1.0
Hashtag Stream	1.5

Table 1: Stream weights for TREC 2012 runs

Listing 1: Building an EPA in Esper

```
create window
rtcount
       .win:time(const_stats_window sec)
       .std:unique(token)
as (token String
    , cnt Long
    , type String
    , ts Long);

insert into rtcount_raw
select
     istream token
    ,count(*) as cnt
    , 'RtCount' as type
    ,current_timestamp()  as ts
from RetweetEvent
                .win:time(
                    const_stats_window sec
                 )
group by token
having count(*) > 0
output  every var_output sec;

insert into rtcount
select istream token
, cnt, 'RtCount' as type
,current_timestamp() as ts
from rtcount_raw s
where cnt > 0;

insert into topKrt
select token,cnt
from rtcount
output  every var_output sec
order by cnt desc limit top_k;
```

For the TREC runs the term frequency *tf* is then being calculated using the following formula:

$$tf_{w,t} = 1 + \sum_{i=1}^{n} \left(TW_{i,t} * \frac{1}{\log_2(rank_t + K)} \right) \qquad (2)$$

with $TW_{i,t}$ being an event stream specific weighting value.

For the runs we use the values shown in table 1. The values are heuristically chosen, based on observations of Twitter, e.g. [19] or [2].

Discussion on real time filtering corpora.

We want to mention that we compressed the text stream for our runs from a *real world 16 day period* to a continuous, one hour data stream, because there is no separate high-volume microblog corpus available for the TREC 2012 filtering task.

This might cause discussion on how representative the results are in comparison to the real world Twitter stream, because in our scenario real world events of 16 days are simulated to happen within one hour. Hence the real Twitter

stream might offer a different density and distribution of terms than the TREC corpus. This will be subject to further research and discussion, but we think that our approach is valid, because experiments with smaller time windows in the post-TREC runs showed also comparable results. Due to space limitation we will postpone this discussion to a later paper.

4.1.2 Determining document collection features in text streams

For determining the document collection based features we employ sliding time windows as well. Sliding time windows were also used for the *BursT* weighting scheme[7], which underpins the viability of our approach. Again, we use the time window to calculate a *streamed inverse document frequency*($sIDF$) value that will be use as the document collection value. The formula is the same as proposed in [6] and explained in [14, p. 504], but it is based on the document count N and term count n in the sliding time window w:

$$sIDF_{w,t} = \log_2 \frac{N_w}{n_w} \tag{3}$$

The final score is then simply calculated by plugging the features into the chosen ranking method. For the TREC runs we used a Vector Space Model and OkapiBM25. For the post-TREC runs we modified OkapiBM25 slightly as well as we used the ATC method[5].

4.2 Pseudo Relevance Feedback for Query Expansion and External Evidence

For the TREC runs no external evidence was used, i.e. no data from Wikipedia or from a search engines were used nor URLs contained in a Tweet were resolved in order to adjust the search profiles. A search profile had the following structure shown in listing 2 and Tweets were only allowed to be considered if they had a tweet id between *querytweetime* and *querynewesttweet*.

Listing 2: Sample search profile TREC filtering trec

```
<top>
 <num>  MB049 </num>
 <title> carbon monoxide law </title>
 <querytime>
            Tue Feb 01 22:44:23 +0000 2011
  </querytime>
 <querytweettime>
            32005451423948800
  </querytweettime>
 <querynewesttweet>
            32569981321347074
  </querynewesttweet>
</top>
```

It has been shown ([11], [8]) that expanding a URL contained in a Tweet can improve the retrieval and ranking performance of algorithms. This approach was not applied here, as the delay between arrival and judgement of a tweet should be kept as small as possible. Hence the resolution of a URL, its parsing and analysis would have introduced an additional time gap that was not acceptable. In an end-user scenario where a real user does not require immediate estimation of new Tweets, this constraint may be loosened and external evidence from a given url might be included. As mentioned above, in this experiment we focus on the immediately available textual information only.

In two TREC-runs pseudo-relevance feedback was included as follows: The system starts without any additional query terms. While the system is running, new Tweets arrive. If the arriving Tweet is considered as being relevant according to the judgements provided by the TREC board, the terms of this Tweet are incorporated into the search profile . The top 5 tokens are added dynamically to the search profile.

But this approach had the drawback that search profiles got dominated by general terms[9] that diluted the search profile. But despite of this one Okapi run submitted to TREC was ranked as #13 out of 69 submitted runs[18].

The relevance feedback mechanism was improved for the post-TREC runs. The positive and negative feedback Tweets were saved in the search profile. During the filtering phase the relevance feedback terms were queried in that way the all - no ranking or selection process - negative feedback terms were subtracted from the positive ones. Only these terms – weighted by factor

$$\alpha$$

– were used in addition to the original query terms. This *substraction* method was the reason for the remarkable increase of the system. In further research we will try to describe in detail why this worked and if it only worked by chance for this scenario.

Without the new feedback method the result stayed around a $T11SU$ around .33, but with the it went up to .47. We also tried a Rocchio based feedback and incorporated decay factor to re-weight the terms in the feedback set. Both did not yield results comparable to the *subtraction* method. The result stayed around a $T11SU$ value of .33.

4.3 Relevance Decision

The relevance decision and thus the setting of the decision threshold are the two main aspects that sustainably influence the performance of an information filtering system.

The relevance decision for the TREC run was as follows. The guidelines of the microblog filtering task asked to provide a retrieval decision for every retrieved Tweet. To do so we simply built an *ideal* document by adding missing search terms to the Tweet under inspection and calculated the score for this Tweet. So we had two scores that could be use to generate a ratio. For the TREC runs we used 0.5 as the threshold, i.e. every Tweet scoring more than 0.5 was marked as relevant.

For the POST TREC runs we used a different approach. Here we exploited a further stream characteristic . We calculated the average score of the positive marked Tweets in the sliding time window. If an incoming Tweet exceeded the average of the positive feedback samples than it was marked relevant. This approach increased all performance measures verifiably.

In order to verify this we did a post-hoc evaluation of our runs and determined the best $T11SU$ value by increasing the threshold step by step. This retrospective approach yielded a fixed threshold and for e.g. *ReverseOkapi* the best $T11SU$ value was at around .46. While precision was almost equal

[9]In [**?** , p. 8] general terms are defined as occurring in positive as well as in negative documents

the dynamic average approach yielded by far better recall values. This is obvious as a dynamic threshold always reflect the current situation of the stream and hence adapts well to changing situations.

5. RUN RESULTS

In this section we present the results for the TREC 2012 and the post-TREC runs. Furthermore we provide a comparison to an *incrementalcorpus* approach based on Lucene[10], i.e. the arriving Tweets were constantly added to the Lucene index and score with the custom Lucene scoring as well as with an custom OkapiBM25 implementation.

5.1 Experimental Setup and Data

The data for the experiments is the TREC 2011 microblog corpus[11]. It is one of the last microblog corpora that is still freely available. Due to its copyright rules Twitter does not allow third parties to provide closed sets of Tweets for research or similar purposes. For example, the *Edinburgh Twitter Corpus* [13] was a scientifically edited corpus but is not available any more due to the aforementioned restrictions.

The TREC 2011 corpus is not a Twitter corpus available for free download either. TREC only offers the id of Tweets that are used for the TREC conference. The Tweets can be downloaded with a tool that crawls either the HTML page of the Tweet with the given ID or downloads a JSON version of the corresponding Tweet. The latter makes use of Twitter API calls which are usually very restricted (e.g. 150 per hour) which dramatically slows down the download process. We have used the HTML version as this allowed us to download the data in a reasonable amount of time.

In total, there are 16 million Tweet ids available. But Twitter is constantly moving its data and that is why it is not assured that each Tweet ID provided by TREC can be downloaded, what in turn has the effect that every research group obtains a different corpus depending on the time Twitter was crawled. We have downloaded the data from Twitter in a time period from October 10 to 19, 2011.

In total we were able to download approx. 12 million Tweets. For our runs, only English Tweets were considered. For this purpose we used the language detection library developed by [17].

We also removed Tweets containing more than four question marks (*'???'*) because there were many Tweets that contained only question marks because of their encoding[12]. Out of the approx. 16 million available ids that were provided by the TREC board we could use approx. 4.5 million. The proceedings of the TREC conference[13] show that this is average of Tweets that could be effectively used.

5.1.1 Evaluation

In total, we have submitted four runs to TREC 2012: Two runs based on OkapiBM25 and two using a Vector Space Model approach, each with and without relevance feedback. After the TREC deadline we continued experimenting. These results are also shown here. We compare our results with the best run from the TREC microblog filtering track in terms of T11SU, f-measure, precision and recall. The runs presented in this section were evaluated against the relevance assessment provided by the TREC board[14]. The qrel[15] value 2 was mapped onto value 1 to get to the binary case. In total 60129 Tweets had been assessed, out of which 116 were considered very bad (-2), 57048 not relevant (0), 2404 relevant (1) and 561 highly relevant(2).

The used evaluation measure are described in [15]. All results are sorted by their T11SU value, which is a utility oriented evaluation measure[4, p. 3] and which is the standard evaluation measure in the filtering tasks of TREC.

Only after submitting the runs to TREC, we have found out that the Vector Space Model (VSM) results got corrupted which became apparent due to their poor performance. The reasons for this are under investigation. The okapiv1 and okapiv2rel performed quite well, while okapiv1 being our best run. This run made it to number 13 out of 69 submitted runs.

In the TREC runs the ones without relevance feedback did better than the ones with relevance feedback. This is due to the naive feedback approach used for the TREC runs 4.2, which diluted the search profile. So the search profiles without feedback stayed more concise and hence performed better.

The naive feedback approach was changed for the post-TREC runs (reverse okapi,atc 2) and this improved the result significantly. Besides changing the feedback mechanism we experimented with different window sizes. Increasing the time window from 10 to 120 seconds also showed better performance[16].

Furthermore the weight for hashtags while determining the local term weight was increased from 1.5 to 12. The ratio between tokens and hashtags is quite skewed, so the hashtag weight would not contribute significantly if we kept the weight so low.

The aforementioned adjustment of the feedback mechanism and the dynamic threshold helped to increase the precision of the post TREC runs. Also the experimentation with different ranking schemes showed interesting result. We tried two versions of Okapi: one with regular document length normalization[17] and one with *reverse* normalization[18]. The goal of the latter was to improve the ranking for longer Tweets. This yielded remarkably better result than *regular* Okapi. Furthermore we tried a classic Vector Space Model approach, as well as an approach (RSV) only based on the document collection features, i.e. only the idf values. Finally in order to show the effectiveness of the event based approach we did some comparison runs based on Lucene. We used Lucene for indexing and retrieving incremental term statistics (document and token count) and calculated a cosine similarity measure as well as a custom

[10] `https://lucene.apache.org/core/`

[11] Access for academic purposes can be requested here `http://trec.nist.gov/data/Tweets/`

[12] Foremost Asian languages were not correctly retrieved by the Crawler

[13] The TREC proceedings will be probably available in the first quarter 2013 `http://trec.nist.gov/proceedings/proceedings.html`

[14] `http://trec.nist.gov/data/microblog/11/microblog11-qrels` access restricted; registration required

[15] Relevance assessment value provided by the TREC board

[16] Due to space reasons we only show the values for the adjusted relevance feedback and changed window size

[17] $norm = \frac{doclength}{averagedoclength}$

[18] $norm = \frac{averagedoclength}{doclength}$

OkapiBM25 one. Both performed considerably worse than the event based approach using sliding windows.

Table 3 shows the concrete numbers of for the experiments conducted after the TREC submission deadline. Table 4 shows the runs without using the subtraction based relevance feedback mechanism.

6. CONCLUSION AND OUTLOOK

Few results submitted to TREC come close to the best value for a specific task, but many of the results are above the median. For the the post-TREC runs the performance was increased. This was foremost due to the adjustment of the window size and the improvement of the relevance feedback mechanism. Both results can be interpreted as a confirmation of the viability of the general approach employing an event processing engine for microblog stream processing. Besides error correction we will focus on the analysis of additional measures of comparison. We will also compare different strategies of construction the temporal corpus, where we will investigate a dynamic adjustment of the window size depending on evaluation metrics like precision, recall or f-measure. Additionally we want investigate how to incorporate the context of the search terms in order to improve the retrieval quality and increase recall. Finally we will contrast sliding windows with incremental approaches and investigate how to efficiently set the decision threshold in order to maximize the performance of the system.

References

[1] Andreas Bauer and Christian Wolff. Event based classification of Web 2.0 text streams. *arXiv.org*, cs.IR, April 2012.

[2] Miles Efron. Hashtag retrieval in a microblogging environment. In *SIGIR '10: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, July 2010.

[3] Opher Etzion and Peter Niblett. *Event Processing in Action*. Manning, 2011.

[4] D A Hull and S Robertson. The TREC-8 filtering track final report. 1999.

[5] R Jin, C Falusos, and A G Hauptmann. Meta-scoring: automatically evaluating term weighting schemes in IR without precision-recall. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 83–89, 2001.

[6] K S Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 60(5):493–502, 2004.

[7] C.H. Lee, C.H. Wu, and T.F. Chien. BursT: A Dynamic Term Weighting Scheme for Mining Microblogging Messages. *Advances in Neural Networks–ISNN 2011*, pages 548–557, 2011.

[8] Feng Liang, Runwei Qiang, and Jianwu Yang. Exploiting real-time information retrieval in the microblogosphere. In *JCDL '12: Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*. ACM Request Permissions, June 2012.

[9] David Luckham. *The Power of Events*. Addison-Wesley, 2005.

[10] K. Mani Chandy and Roy Schulte. *Event Processing. Designing IT Systems for Agile Companies*. McGraw Hill, 2010.

[11] K Massoudi, M Tsagkias, M de Rijke, and W Weerkamp. Incorporating query expansion and quality indicators in searching microblog posts. *Advances in Information Retrieval*, pages 362–367, 2011.

[12] N. Naveed, T. Gottron, J. Kunegis, and A.C. Alhadi. Bad News Travel Fast: A Content-based Analysis of Interestingness on Twitter. *websci11.org*, 2011.

[13] Saša Petrović, Miles Osborne, and Victor Lavrenko. The Edinburgh Twitter corpus. In *WSA '10: Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*. Association for Computational Linguistics, June 2010.

[14] S Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520, 2004.

[15] S Robertson and I Soboroff. The TREC 2002 filtering track report. 2002.

[16] G Salton and C Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[17] Nakatani Shuyo. Language detection library for java, 2010. URL `http://code.google.com/p/language-detection/`.

[18] I Soboroff, I Ounis, and Jimmy Lin. Overview of the TREC-2012 Microblog Track. In *trec.nist.gov*. NIST.

[19] Michael J Welch, Uri Schonfeld, Dan He, and Junghoo Cho. Topical semantics of twitter links. In *WSDM '11: Proceedings of the fourth ACM international conference on Web search and data mining*. ACM Request Permissions, February 2011.

Run id	Precision	Recall	F-Measure@.5	T11SU	Decision Threshold
okapiv1 (TREC)	0.3370	0.1024	0.3338	0.1916	0.5
okapiv2rel (TREC)	0.2831	0.1486	0.2978	0.1942	0.5
vsmv1 (TREC)	0.1217	0.0732	0.2690	0.0616	0.5
vmsv2rel (TREC)	0.1411	0.0518	0.381	0.0835	0.5

Table 2: TREC run summary

	scoring_function	prec	recall	f_measure	t11su	accuracy	specificity	tp	tn	fp	fn
1	VSM	0.4074	0.4221	0.3783	0.3391	0.8945	0.9089	771	30191	3026	625
2	tfidf	0.6199	0.2115	0.3737	0.4155	0.9275	0.9541	404	15165	729	488
3	RSV	0.6060	0.3348	0.4764	0.4477	0.9559	0.9786	568	32418	710	813
4	reverse_okapi	0.7198	0.3482	0.5518	0.5148	0.9677	0.9904	594	32916	318	802
5	okapi_stream_count	0.5977	0.3437	0.4775	0.4504	0.9557	0.9783	583	32515	720	813
6	atc	0.4861	0.3438	0.4188	0.3898	0.9470	0.9641	675	31908	1189	634
7	lucene_tfidf_incr	0.3514	0.1364	0.2207	0.3027	0.8761	0.9053	307	14392	1506	573
8	lucene_okapi	0.3288	0.1953	0.2274	0.2824	0.6833	0.6941	430	11035	4863	450

Table 3: Post TReC runs using relevance feedback – summary

	scoring_function	prec	recall	f_measure	t11su	accuracy	specificity	sum(tp)	sum(tn)	sum(fp)	sum(fn)
1	atc	0.2725	0.1819	0.2015	0.2727	0.5122	1	448	2082	1968	441
3	okapi_stream_count	0.2726	0.3282	0.2413	0.2469	0.7055	1	821	9114	3658	489
4	reverse_okapi	0.3026	0.3483	0.2711	0.2681	0.7595	1	801	9893	2878	509
5	RSV	0.3497	0.1717	0.2465	0.3051	0.7143	1	444	3084	966	445
6	tfidf	0.3227	0.1890	0.2295	0.2780	0.5868	1	518	2380	1670	371
7	VSM	0.3150	0.1667	0.2026	0.2719	0.5386	1	407	2253	1797	482

Table 4: Post TReC runs without relevance feedback – summary