RMIT at TREC 2011 Microblog Track

Matthias Petri
RMIT University
School of CS&IT
matthias.petri@rmit.edu.au

J. Shane Culpepper
RMIT University
School of CS&IT
shane.culpepper@rmit.edu.au

Falk Scholer
RMIT University
School of CS&IT
falk.scholer@rmit.edu.au

***Abstract*—This paper describes our submission to the TREC 2011 microblog task. For the experiments, we use our new self-index search engine, NeWT, to support ranked search in the Twitter document corpus. We use a combination of phrase queries and degrading conjunctive Boolean intersection to improve retrieval effectiveness.**

***Keywords*-self-index; full-text search, phrases, threshold; intersection**

I. Introduction

Twitter is a relatively new medium. Tweets, the document unit of Twitter, are restricted by length to 140 characters, but can contain hyperlinks to specific topics or users. Communication using tweets is similar to text communication over mobile phones, which became popular at the turn of the century. Messages contain abbreviations, small talk, and conversational text. Additionally, Twitter has become a global phenomenon whereby users from around the world communicate with each other, leading to a truly multilingual collection. These aspects make searching using traditional search methods difficult. Normalizing multilingual conversational text mixed SMS text-speak is particularly problematic.

In our microblog submission, we mitigate these problems by using our character-aligned self-index, NeWT, for storage and retrieval. Self-indexes are traditionally used to provide full-text search in the bioinformatics domain. In this work, we use NeWT to perform Boolean conjunctive phrase queries over the multilingual Twitter corpus, without the need to create an English language specific index. The phrase queries are created automatically or manually for the given topics, and were submitted as individual runs.

The paper is structured as follows: First, we give a brief introduction to self-indexes. We then discuss the preprocessing steps for the queries and the collection. Next, we describe our different approaches, and evaluate our submissions using the baseline provided by the TREC organizers, as well as the median score over all participants for each topic.

II. Self-Indexes

A *suffix array* stores positions of suffixes of a given text T of size n in lexicographical order, and can be used to find all occurrences of pattern P of length m in time proportional to m. Unfortunately, suffix arrays require $4n$ bytes of space in addition to the space required to store T. In contrast, a *self-index* can provide the same search functionality as a suffix array while only requiring space roughly equal to the size of the compressed representation of T, and do not require the text T to be stored explicitly [1].

Muthukrishnan first proposed the use of a suffix array to support document level retrieval [2]. Recently, self-indexes have been augmented to provide document level retrieval as well as simple TF·IDF relevance ranking [3]. In addition to relevance ranking, other techniques such as set intersection or top-k ranked retrieval can be efficiently performed using a self-index [4]. For more information on self-indexes, refer to the surveys by Navarro and Mäkinen [5] and Ferragina et al. [6].

III. Preprocessing

In this section we discuss preprocessing of both the collection and the query set for our submitted runs.

A. Collection

When constructing an inverted index, each document is parsed into tokens and then represented as a sequence of terms. Normalization usually includes techniques such as downcasing, stemming and stopping. Determining the terms that are added to the dictionary becomes even more complicated in a multilingual environment such as Twitter. Morphological parsing is often language specific, and the problem can be exacerbated by mixing languages and SMS text-speak is a single collection.

Alternatively, self-indexes are character based. Therefore, extracting terms from each document is not required. To simplify query processing, the only preprocessing necessary is to downcase Latin symbols. No additional preprocessing steps to support multilingual queries are necessary as linguistic choices can be made at query time.

B. Queries

Self-indexes are not term based, and the time required to process a query depends only on the total query length. This implies that, in practice, the cost to process a phrase query is the same as the cost of processing a bag of words query. Therefore, we focus mainly on different approaches to query processing for each of our run submissions. We use one automatic and three manual approaches to extract phrases from the given query terms. The different approaches are described below:

RMITAR (automatic): No phrases were extracted from the given query terms. Query terms with only 3 or less symbols were enclosed using space symbols to prevent substring matches within words. For example, 'nsa' becomes '␣nsa␣' so 'unsafe' is not matched. **RMITM/RMITMR/RMITMRR (manual):** Similar to the automatic approach, we again enclose words with potential substring matches in unrelated words with spaces. Additionally, phrase queries were created for:

1. Proper names. For example "oprah winfrey".
2. Combination of terms to disambiguate meaning. For example "birth certificate".
3. Combination of query terms. E.g. "moscow airport" and "moscow airport bomb"

In addition to creating phrases we also manually stemmed query terms to match derived forms of a query term. E.g. "recovery" becomes "recover".

IV. Query processing

Document relevance is determined threshold based intersection. For each topic, we perform multiple Boolean conjunctive queries until a predefined number of documents t is retrieved, or we process all query terms. Note a query term may be a single English word or a short phrase, depending on the query method used.

For a topic with k query terms, we intersect the results of each query with an intersection threshold of k. This is a classic conjunctive Boolean query where all terms must occur in a returned document. Next, we perform the same intersection with an intersection threshold of $k-1$, adding all "new" documents to the result set. The order of each document in the final result list is determined by the threshold k with which it was retrieved in first. We continue the intersection up to a threshold of $k = 2$. If the result set contains less than t documents, we search the collection for each individual query term in ascending order of their frequency in the collection until we retrieve t documents or all query terms have been processed.

After retrieving up to t relevant documents we reorder based on query time and discard all "future" tweets. We further filter matching non English tweets using an English word dictionary. We used a standard dictionary and the training queries to remove words that occur in non-English tweets. To filter tweets, we require 50% of the words in a tweet to exist in the dictionary. The threshold for the automatic run (RMITAR) and RMITMRR was set to $t = 200$. For RMITM and RMITMR the threshold was $t = 30$ and $t = 500$ respectively.

V. External and future resources

During query processing we reorder query terms based on their frequency in the collection. This frequency is based on the frequency of the term in the complete collection and not just before query time. The English word dictionary to filter non-English tweets is an external resource.

VI. Evaluation

We first compare our different approaches before comparing against the baseline provided by the TREC organizers, and the median performance of all participants. We then briefly discuss topic specific performance issues we encountered during our evaluation.

A. Run comparison

	High			All		
Runs	MAP	R-prec	P@30	MAP	R-prec	P@30
RMITAR	0.1672	0.1728	0.0788	0.2382	0.2946	0.2211
RMITM	0.1619	0.1778	0.0970	**0.2430**	**0.3046**	0.2721
RMITMR	0.1649	0.1728	0.0727	0.2320	0.2842	0.2041
RMITMRR	**0.1716**	**0.1904**	**0.0980**	0.2345	0.2830	**0.3163**

Table I
Average MAP, R-prec and P@30 scores over all submitted runs for highly relevant (High) and all relevant (All) tweets.

Table I lists average MAP,R-prec and P@30 for our submitted runs. Note that the manual runs perform best for all metrics. The automatic run performs close to all manual runs. Indeed there is no statistically significant difference (paired t-test, $p > 0.1$) between the best manual run (RMITMRR) and the automatic run (RMITAR).

Manual construction of phrase query terms appears to add only a marginal gain in overall effectiveness in our experimental results. Each individual document is very short. Therefore, conjunctive queries have a similar effect on the returned documents as phrase queries. Note that we do not use any relevance ranking metric (e.g. BM25). We also do not normalize by document length as IDL has little impact on the Twitter corpus where all documents are of roughly equal length.

B. Baseline comparison

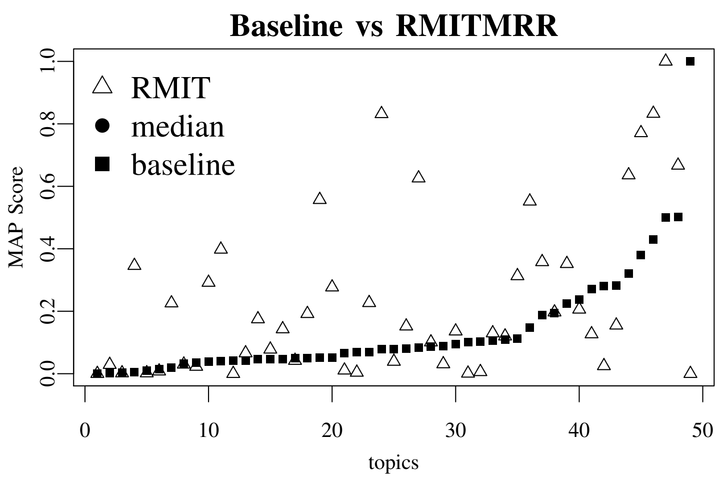
	High			All		
Runs	MAP	R-prec	P@30	MAP	R-prec	P@30
DISJ-BASE	N/A	N/A	N/A	0.1411	0.1827	0.0986
MED-AVG	0.1649	0.1728	0.0727	0.1433	0.6149	N/A
RMITMRR	**0.1716**	**0.1904**	**0.0980**	**0.2345**	**0.2830**	**0.3163**

Table II
Average MAP, R-prec and P@30 scores over all submitted runs for highly relevant (High) and all relevant (All) tweets.

We now compare our best run (RMITMRR) to the disjunctive baseline run provided by the TREC organizers, as well as the median reported scores. In Table II, RMITMRR outperforms both the provided baseline, as well as the median score over all participants. Overall we significantly outperformed both the provided baseline (p-value = 0.0433) and the median average score submitted by all participants (p-value = 0.0425). The results were obtained using a paired t-test.

C. Topic evaluation

We now evaluate each topic individually. Figure 1 shows a topic level comparison of our RMITMRR run to the provided disjunctive baseline, and the median score



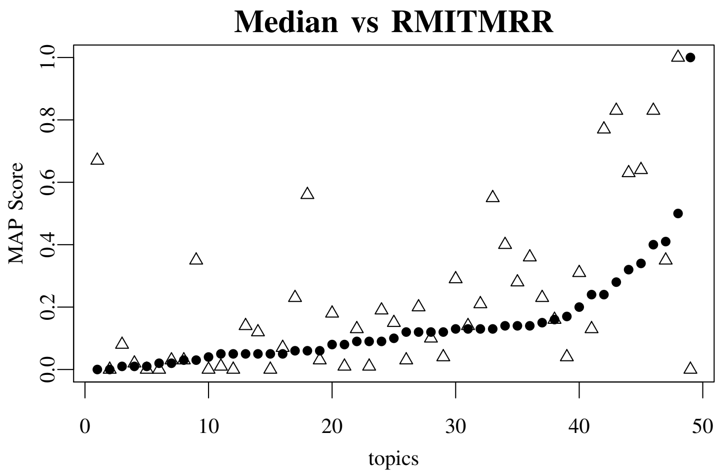


Figure 1. Topic level evaluation by comparing RMITMRR to both the provided baseline and the median scores over all participants. In both graphs the topics are sorted in increasing order of MAP score of the run as compared to RMITMRR.

reported by all participants. In both graphs, the topics are sorted in increasing order of MAP score of the run as compared to RMITMRR. In 15 out of 49 topics, we perform worse than the median score achieved by all participants. For 12 out of 49 topics, we perform worse than the provided disjunctive baseline. The problematic topics are listed in Table III. Most of the problems can be attributed to the following:

1. Substring matching in non-relevant words. Some query words match substrings in unrelated words. For example: "nist" matched "mi<u>nist</u>er". This could have been avoided if all query terms were encapsulated in spaces as described above.
2. Intersection does not return enough results. If the degrading threshold based intersection does not return t results, each query word is processed individually based on frequency in the collection. This leads to non-relevant tweets close the query-time being included in the result set.
3. Retweets. We did consider approaches to filter manual retweets.
4. Matching of query terms in unrelated hashtags or usernames. Unencapsulated query terms also matched substrings in hashtags or usernames which are not related to the query topic. If the matched hashtag or username occurs often in the collection, the result set is polluted with false matches.

Topic	Query
MB004	mexico , drug war
MB005	nist, computer, security
MB008	phone hacking, british
MB010	egyptian, protester, attack, museum
MB013	oprah winfrey, oprah, half-sister
MB017	white stripe, breakup
MB019	cuomo, budget cut
MB020	taco bell, filling, lawsuit
MB021	emanuel, residency ,court , ruling
MB037	gifford, recover
MB038	protest, jordan
MB044	white house , spokesman , replace
MB049	carbon monoxide , law , carbon

Table III
PROBLEMATIC TOPICS FOR SIMPLE SELF-INDEXING APPROACHES.

VII. Conclusion

We show that self-indexes are a promising approach for microblog retrieval tasks. The submitted runs outperform the provided baseline and the median participant score significantly. Our approach requires little or no linguistic preprocessing at index time, and offers greater flexibility for query-time processing. However, leveraging query processing in self-indexes to improve retrieval effectiveness by combining phrase and bag-of-word queries remains a topic of future work.

References

[1] P. Ferragina and G. Manzini, "Opportunistic data structures with applications." in *Proceedings of the 41st IEEE Annual Symposium on Foundatations of Computer Science (FOCS 2000)*. IEEE Computer Society Press, November 2000, pp. 390–398.

[2] S. Muthukrishnan, "Efficient algorithms for document retrieval problems," in *Proceedings of the 13th ACM-SIAM symposium on Discrete algorithms (SODA 2002)*. ACM/SIAM, January 2002, pp. 657–666.

[3] K. Sadakane, "Succinct data structures for flexible text retrieval systems." *Journal of Discrete Algorithms*, vol. 5, no. 1, pp. 12–22, March 2007.

[4] J. S. Culpepper, G. Navarro, S. J. Puglisi, and A. Turpin, "Top-k ranked document search in general text databases." in *Proceedings of the 18th Annual European Symposium on Algorithms (ESA 2010), Part II*, ser. LNCS, M. de Berg and U. Meyer, Eds., vol. 6347. Springer, 2010, pp. 194–205.

[5] G. Navarro and V. Mäkinen, "Compressed full-text indexes." *ACM Computing Surveys*, vol. 39, no. 1, pp. 2–1 – 2–61, 2007.

[6] P. Ferragina, R. González, G. Navarro, and R. Venturini, "Compressed text indexes: from theory to practice," *Journal of Experimental Algorithmics*, vol. 13, pp. 1.12–1.31, 2009.