The Impact of Future Term Statistics in Real-Time Tweet Search

Yulu Wang[1,2] and Jimmy Lin[1,2,3]

[1] Dept. of Computer Science,
[2] Institute for Advanced Computer Studies, [3] The iSchool
University of Maryland, College Park, USA
ylwang@cs.umd.edu, jimmylin@umd.edu

Abstract. In the real-time tweet search task operationalized in the TREC Microblog evaluations, a topic consists of a query Q and a time t, modeling the task where the user wishes to see the most recent but relevant tweets that address the information need. To simulate the real-time aspect of the task in an evaluation setting, many systems search over the entire collection and then discard results that occur after the query time. This approach, while computationally efficient, "cheats" in that it takes advantage of term statistics from documents not available at query time (i.e., future information). We show, however, that such results are nearly identical to a "gold standard" method that builds a separate index for each topic containing only those documents that occur before the query time. The implications of this finding on evaluation, system design, and user task models are discussed.

1 Introduction

In this paper we tackle a simple but substantive question: do term statistics from future documents impact retrieval effectiveness in the real-time tweet search task? The answer to this question holds important implications for the design of retrieval systems and how they are evaluated. We explore this question in the context of the TREC 2011 and 2012 Microblog evaluations and conclude that simple term statistics from future documents (i.e., tweets posted after the query time) do not have a significant impact on retrieval effectiveness.

We adopt the definition of the real-time tweet search task operationalized in TREC [1]. A topic consists of a query Q and a time t, modeling the task where the user wishes to see the most recent but relevant tweets that address the information need. Operationally, the "real-time" nature is *simulated* in the evaluation, as participants are able to acquire the entire collection of tweets all at once. One common approach is to treat the problem as standard *ad hoc* retrieval over the *entire* collection, and then filter any hits that occur after the query time. This widely-adopted approach, however, violates the real-time restriction, because by performing retrieval over the entire collection, the model gains access to future information—for example, term statistics from documents that were created after the query time. To meet the real-time criterion, participants must

M. de Rijke et al. (Eds.): ECIR 2014, LNCS 8416, pp. 567–572, 2014.

ensure that all features used in the ranking model are available at the time of the query. The easiest way to accomplish this is to build a separate inverted index for each topic, and in each index ingest only those tweets that appear before the query time. Needless to say, this "gold standard' approach is awkward, time consuming, and impractical for large collections.

Our study asks a simple question: does it matter? If we compare the "search the entire collection and filter" approach vs. the "one index per topic" approach, are the results different? In other words, does information about future documents such as term statistics affect retrieval effectiveness? We experimentally compare these two approaches and conclude that the answer is *no*—results from both are nearly identical. This finding has implications for system and evaluation design: we can adopt the much simpler approach without "cheating".

2 Methodology

Our study used test collections created from the recent Microblog tracks at TREC [1,2]. The 2011 and 2012 evaluations used the Tweets2011 collection, which consists of, after some spam removal, an approximately 1% sample of tweets from January 23, 2011 to February 7, 2011 (inclusive), totaling about 16 million tweets. Major events that took place within this time frame include the massive democracy demonstrations in Egypt as well as the Super Bowl in the United States. There are 49 topics for TREC 2011[1] and 60 topics for TREC 2012. Each topic consists of a query and an associated timestamp, which indicates when the query was issued. Using a standard pooling strategy, NIST assessors evaluated a total of 114K tweets and assigned one of four judgments to each: spam, not relevant, relevant, and highly-relevant. For the purpose of our experiments, we considered both relevant and highly-relevant tweets "relevant".

Our experiment compared two conditions: In the "complete index" condition, we built a standard inverted index over all 16 million tweets. In the "per-topic index" condition, we built 108 separate indexes, one for each topic. Each index contains only tweets that were created before the query time, making it impossible to inadvertently use future information not available at query time. For all experiments we used the open-source tools that were released as part of the TREC 2013 Microblog evaluation,[2] which are based on the Lucene search engine. For retrieval we used the query-likelihood implementation in Lucene, which serves as the baseline for the 2013 evaluation.

In the complete index condition, the query associated with each topic was treated as a bag of words and run over the index for the entire collection; tweets after the query time were then discarded. In the per-topic index condition, each query was run over its dedicated index. The top 1000 hits in both cases were evaluated. We compared mean average precision (MAP) and precision at rank cutoff 30 (P30), which were the two metrics used in the TREC evaluations.

[1] One topic had no relevant documents and was discarded from our analyses.

[2] `http://twittertools.cc/`

Table 1. Comparing indexing approaches on TREC 2011 and 2012 topics

	TREC 2011		**TREC 2012**	
Condition	**MAP**	**P30**	**MAP**	**P30**
Complete index	0.3042	0.3476	0.1818	0.2932
One index per topic	0.3062	0.3497	0.1816	0.2932

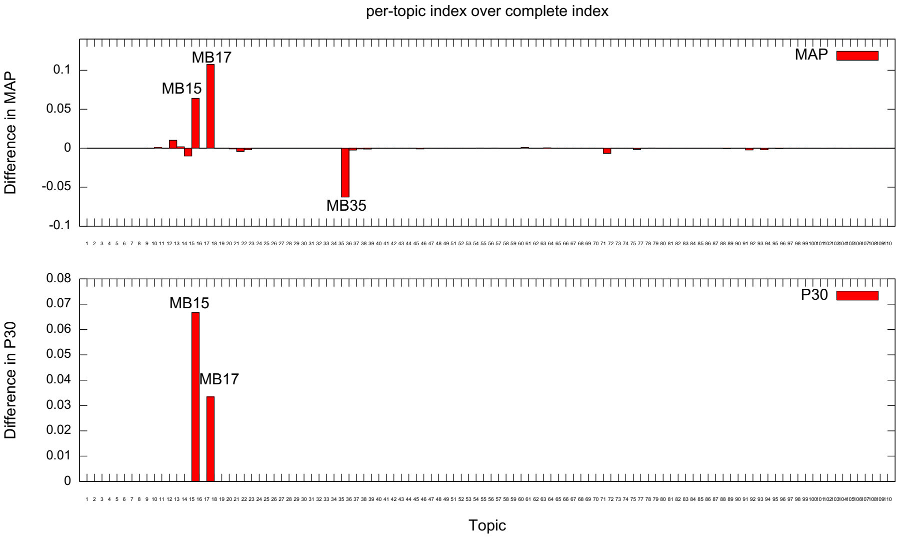

Fig. 1. Per-topic absolute differences in MAP (top) and P30 (bottom) for TREC topics. Bars above the origin denote cases where the per-topic index is more effective.

3 Experimental Results

Our evaluation results are shown in Table 1, comparing the “complete index” approach vs. the “one index per topic” approach. Numbers for the TREC 2011 and 2012 topics are shown separately. Per-topic absolute differences between the two experimental conditions are presented in Figure 1. The x-axis shows each topic, with sets of bars denoting absolute differences in MAP and P30. Across the 2011 and 2012 test collections, if we measure by MAP, the effectiveness of both approaches is identical for 77 topics; the effectiveness is better using the “one index per topic” approach for 14 topics and the effectiveness of the “complete index” approach is better for 17 topics. If we measure by P30, the effectiveness of the two approaches is identical for 106 topics; for the remaining two topics the “one index per topic” approach is better. The overall differences between the two approaches are not statistically significant.

For each topic, we examined the inverse document frequency (*idf*) of the query terms in both the complete index and per-topic index.[3] We found that the *idf* in both conditions were nearly identical across all query terms—this is not surprising considering that *idf* is on a log scale, and it takes substantial variations in document frequencies to have a noticeable affect on the value. However, Figure 1 shows that there is a large difference in effectiveness for a few topics: MB15, MB17, and MB35. We explore these topics below.

For topic MB35, the query is "Sargent Shriver tributes". There are substantial differences in *idf* between the complete index and per-topic index for "Sargent" and "Shriver", but not for "tributes". We traced these differences to the query time, which is 07:18 on January 24—note that the corpus begins on January 23, which means that the per-topic index is rather small. In such a small collection, the term statistics can be idiosyncratic and not reflective of term distributions in tweets. Thus, the per-topic index MAP is substantially lower than the complete index MAP. However, in both conditions the same number of relevant tweets were retrieved in the top 30 hits.

Topic MB15, "Thorpe return in 2012 Olympics", asks about the Australian swimmer. The *idf* for the term "thorp" (stemmed) is higher in the per-topic index than in the complete index—this has the effect of emphasizing the importance of the swimmer's name, which leads to higher effectiveness. The *idf* of the other terms are nearly identical. By focusing on a smaller slice of tweets, the per-topic index contains relatively more mentions of the name. Note that it is *not* generally the case that queries with relatively rare terms such as person names give rise to effectiveness differences between the complete index and per-topic index conditions. The TREC datasets contain many person names but no others exhibit the effectiveness difference we observe in MB15.

Topic MB17, "White Stripes breakup", asks about The White Stripes, an American rock duo. The term "stripe" has a higher *idf* in the per-topic index than the complete index; the other terms have nearly identical *idf* values. Since the name of the group consists of common words in English, the effectiveness difference appears to be idiosyncratic.

It is well known that term occurrences can change rapidly on Twitter—this is the basis of Twitter trends. However, most rapid increases in term occurrences are relatively short lived and correspond to "what's hot" at the moment (e.g., a breaking news story, a TV show, a sports contest, etc.). For the most part, the global Twitter conversation shifts to some other topic within a short span of time [4]. These bursty behaviors do not appear to contain sufficient "mass" to substantially impact term statistics such as *idf*, which has built in "damping" since it is on a log scale. Furthermore, changes in *idf* values alone are not sufficient to yield different rank orders in retrieved hits—if all the query terms become more frequent to the same extent, the document scores would change but not

[3] We are aware that collection frequency, rather than document frequency, is used in query likelihood. However, i) *idf* is a general concept that applies broadly to other scoring models, and ii) previous work has confirmed that *df* and *cf* are nearly identical for Twitter search since terms almost always have $tf = 1$ in tweets [3].

the ranking. Since user queries often ask about a coherent concept or terms that are otherwise correlated, changes in term statistics are often correlated as well. To yield a different rank order of results, two conditions must be met: i) the term statistics must change enough to have an impact on a log scale, and ii) the relationship between the term statistics must be different. We have empirically shown that these conditions are rarely met in practice, and thus results from the complete index and per-topic indexes are mostly indistinguishable.

4 Discussion

One potential objection to these results is that we use a simple baseline query-likelihood model for comparing the alternative approaches. We justify this setup by considering modern search architectures, which typically break ranking into three stages: candidate generation, feature extraction, and document reranking. There are two main reasons for this multi-stage design. First, there is a general consensus that learning to rank provides the best solution to document ranking [5]. As it is difficult to apply machine-learned models over the *entire* collection, in practice a candidate list of potentially-relevant documents is first generated. Thus, learning to rank is actually a *reranking* problem (hence the first and third stages). Second, separating candidate generation from feature extraction has the advantage of providing better control over cost/quality tradeoffs. In this architecture, our experiments consider the candidate generation stage. Additional work has shown that end-to-end retrieval effectiveness is insensitive to the candidate generation algorithm [6,3], which means that our experiments using simple query-likelihood accurately reflect real-world conditions.

Examples of this search architecture abound in industry [7] and academia [8,6]. In fact, the TREC 2013 Microblog evaluation is exactly set up along these lines: participants do not have access to the raw collection—instead, they must complete the task via a search API that returns candidate results. Note that in the feature extraction and reranking stages, features manipulated by a system are properties of the candidate documents (which are guaranteed to occur before the query time by filtering). Thus, it is more difficult to inadvertently include future information not available at query time.

Why is our experimental finding important? These results show that for evaluation purposes, we can deploy a *single* complete index to *simulate* the real-time search task, provided that we filter results that occur after the query time. It is not necessary to build a separate index for each topic, which is cumbersome, time-intensive, space-wasting, and simply impractical for large collections.

Finally, our results hold implications for user task models in searching tweet collections. Consider a scenario where a journalist is investigating a sports scandal that has been brewing for the past several weeks. She just got news of a breaking development, and turns to searching tweets to find out more details. Since this particular news story has been developing for several weeks, any keyword search involving the athlete's name might bring up results from many different points in time. It would be desirable if the journalist could specify that she is only interested in the most recent tweets. This is the real-time search scenario we have been considering. However, consider another scenario in which a journalist is searching an archive of tweets as part of a retrospective piece on the impact of social media on the course of the Egyptian revolution. Let's say she is particularly interested in activists using Twitter for "on the ground" reporting purposes. In this case, the journalist has a concrete idea when the relevant tweets should occur (e.g., when protesters were gathered in Tahrir Square) and would like to include this external knowledge as a query time.

These are two distinct task models and they roughly correspond to the per-topic index and complete index conditions, respectively. Our results suggest that a single backend might be sufficient for developing systems that are specialized to either task. This does not necessarily mean that we can use the same algorithms—simply that they might be able to share the same infrastructure, thus simplifying development.

5 Conclusions

We examine a question that arose from recent TREC Microblog evaluations—whether the computational expedient of searching the entire collection with post-hoc filtering unfairly takes advantage of future information. The answer is *no*, which validates the empirical approach that many researchers had been taking all along. Now we affirm that they really weren't "cheating". Whew!

Acknowledgments. This work has been supported by NSF under awards IIS-1144034 and IIS-1218043. Any opinions, findings, or conclusions are the authors' and do not necessarily reflect those of the sponsor. The second author is grateful to Esther for her loving support and dedicates this work to Joshua and Jacob.

References

1. Ounis, I., Macdonald, C., Lin, J., Soboroff, I.: Overview of the TREC-2011 Microblog Track. In: TREC (2011)
2. Soboroff, I., Ounis, I., Macdonald, C., Lin, J.: Overview of the TREC-2012 Microblog Track. In: TREC (2012)
3. Asadi, N., Lin, J.: Fast candidate generation for real-time tweet search with Bloom filter chains. ACM Transactions on Information Systems 31(3), article 13 (2013)
4. Lin, J., Mishne, G.: A study of "churn" in tweets and real-time search queries. In: ICWSM, pp. 503–506 (2012)
5. Li, H.: Learning to Rank for Information Retrieval and Natural Language Processing. Morgan & Claypool Publishers (2011)
6. Asadi, N., Lin, J.: Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In: SIGIR, pp. 997–1000 (2013)
7. Cambazoglu, B.B., Zaragoza, H., Chapelle, O., Chen, J., Liao, C., Zheng, Z., Degenhardt, J.: Early exit optimizations for additive machine learned ranking systems. In: WSDM, pp. 411–420 (2010)
8. Macdonald, C., Santos, R.L., Ounis, I.: The whens and hows of learning to rank for web search. Information Retrieval 16(5), 584–628 (2013)