Information Processing and Management 51 (2015) 89–113

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Burst-aware data fusion for microblog search

Shangsong Liang *, Maarten de Rijke

University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 26 March 2014
Received in revised form 31 October 2014
Accepted 31 October 2014

Keywords:
Information retrieval
Microblog search
Rank aggregation
Burst detection
Temporal information retrieval

ABSTRACT

We consider the problem of searching posts in microblog environments. We frame this microblog post search problem as a late data fusion problem. Previous work on data fusion has mainly focused on aggregating document lists based on retrieval status values or ranks of documents without fully utilizing temporal features of the set of documents being fused. Additionally, previous work on data fusion has often worked on the assumption that only documents that are highly ranked in many of the lists are likely to be of relevance. We propose BurstFuseX, a fusion model that not only utilizes a microblog post's ranking information but also exploits its publication time. BurstFuseX builds on an existing fusion method and rewards posts that are published in or near a burst of posts that are highly ranked in many of the lists being aggregated. We experimentally verify the effectiveness of the proposed late data fusion algorithm, and demonstrate that in terms of mean average precision it significantly outperforms the standard, state-of-the-art fusion approaches as well as burst or time-sensitive retrieval methods.

1. Introduction

Microblogging platforms, such as Twitter,[1] have become indispensable communication channels through which hundreds of millions of users around the world witness breaking news events. The characteristics of the posts, such as their limited length, along with easy access on many platforms, facilitate regular status updates by large numbers of people (Zhao & Rosson, 2009). Microblogging platforms display fast paced dynamics as reflected by rapidly evolving topics (Yang & Leskovec, 2011). Searching posts in such rapidly changing environments is a challenge (Ounis, Macdonald, Lin, & Soboroff, 2011). To tackle this problem, much previous work has focused on content-based criteria for ranking posts in response to a query, in combination with a broad range of other ranking criteria, including, e.g., existence of hyperlinks, hashtags and retweets.

Fusion is a popular method for generating result lists based on multiple ranking criteria. Previous research has found that data fusion can enhance the retrieval performance in many cases (Dong & Srivastava, 2013; Shaw, Fox, Shaw, & Fox, 1994; Wu, 2012). In this paper, we look at the problem of searching microblog posts as a late data fusion task (Shaw et al., 1994): we fuse ranked lists of posts produced by a diverse set of microblog post rankers into a single final ranked list of posts. In the following, we consider the case where only ranks and publication times are available and no other additional information is provided such as the retrieval status values or the contents of the posts. We focus on a particular microblog search scenario, one that was studied at the Text REtrieval Conference (TREC) 2011 and 2012 Microblog tracks (Ounis et al., 2011; Soboroff,

* Corresponding author.
E-mail addresses: s.liang@uva.nl (S. Liang), derijke@uva.nl (M. de Rijke).

[1] http://www.twitter.com.

http://dx.doi.org/10.1016/j.ipm.2014.10.008
0306-4573/

Ounis, Macdonald, & Lin, 2012). The task uses Twitter data and is defined as follows: given a query with a timestamp, return relevant and interesting tweets.

Fusing multiple document lists that have been retrieved from a corpus in response to a query so as to compile a single result list, has a long history (Kozorovitsky & Kurland, 2011; Shaw et al., 1994; Tsai, Wang, & Chen, 2008), with the CombSUM family (CombMax, CombMin, CombSUM, CombANZ, CombMNX, CombMNZ, etc. Lee, 1995) of fusion methods being the oldest and one of the most successful ones for many IR tasks (He & Wu, 2008; Sheldon, Shokouhi, Szummer, & Craswell, 2011; Tsagkias, de Rijke, & Weerkamp, 2011; Tsai et al., 2008). The lists are often produced by multiple ranking functions, e.g., query representations or document representations (Croft, 2000). Many effective fusion methods are based on the assumption that only documents that are highly ranked in many of the lists are likely to be relevant (Aslam & Montague, 2001; Croft, 2000; Dwork, Kumar, Naor, & Sivakumar, 2001; Kozorovitsky & Kurland, 2011; Lee, 1995; Montague & Aslam, 2002; Shaw et al., 1994; Tsagkias et al., 2011). As a consequence, a relevant document will be ranked low in the final fused list if it appears only in a single list and is ranked low in this list.

The characteristics of microblog environments suggest a different perspective. In such environments news events trigger people to talk about the topics related to an event mostly during specific short time intervals (Chen, Chen, Zhang, Wang, & Bu, 2010; Hoonlor, Szymanski, Zaki, & Chaoji, 2012; Lappas, Arai, Platakis, Kotsakos, & Gunopulos, 2009; Mathioudakis, Bansal, & Koudas, 2010; Peetz, Meij, de Rijke, & Weerkamp, 2012; Vlachos, Meek, & Vagena, 2004). For instance, people talked about the "2014 Eastern Synchronized Skating Sectional Championship" mainly between January 30 and February 1, 2014, which is when the championship was held. Posts created before the beginning or after the ending of the event are less likely to discuss the championship competitions and, hence, are less likely to be relevant. This observation leads to the following intuition about fusing ranked lists of microblog posts. If a post d and (other) relevant posts $d_1, \ldots, d_k$ were published within the same narrow time window, and the relevant posts $d_1, \ldots, d_k$ are ranked highly in many of the lists to be merged, then post d should be "rewarded" by boosting its rank, even if, in the extreme case, it appears in only one list where it is ranked low. Fig. 1 illustrates this intuition; there, post d_2 is ranked low in list L_1 but our intuition suggests that it should be rewarded as it was published in the same narrow time window in which a large number of posts occur that are ranked high in many lists; in contrast, d_8, while ranked high in L_m, receives no such bonus as it was published outside the narrow window.

To tackle the problem of microblog post search, we propose BurstFuseX, a novel probabilistic model that not only utilizes information traditionally used when merging ranked lists, such as ranks, but also exploits temporal information, i.e., the publication timestamps of microblog posts. In our fusion model, we focus on the case where only ranks and publication timestamps are available and no additional information is provided—such as the content of the posts, the post's RSVs (Relevance Status Values), and the resources the posts link to. In fact, accessing the contents of posts may be inefficient and hence inappropriate in dynamic environments such as microblog search. In addition, the content may not be available in all scenarios (Salakhutdinov & Mnih, 2008). Briefly, BurstFuseX first calls a standard document fusion method X to merge a set of ranked lists of microblog posts for a given query. Subsequently, as illustrated in Fig. 1, based on the fused scores produced by method X, we detect windows of timestamps of high-scoring posts. These windows give rise to bursts of posts. We then reward posts that are published in the temporal vicinity of a burst that contains high-scoring posts.

In our experiments aimed at assessing the performance of BurstFuseX, we sample runs that have been submitted to the TREC 2011 and 2012 Microblog tracks and fuse them using BurstFuseX, respectively. For the underlying fusion method X (on top of which BurstFuseX builds), we consider three alternatives: two unsupervised fusion methods, CombSUM (Shaw et al.,

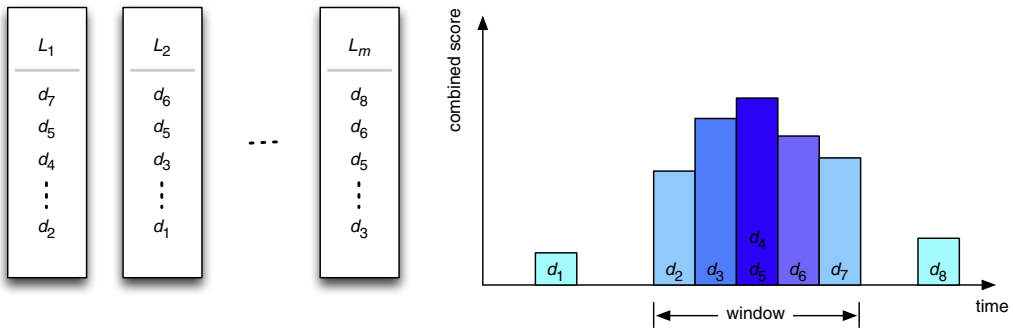

Fig. 1. Rewarding posts that are published in the same narrow time frame as a large number of (supposedly) relevant posts. On the left, we display m ranked lists of posts that were produced in response to a given query; these lists need to be fused. Post d_2 only occurs in list L_1 and it is ranked low in L_2; d_8 also occurs in a single list, L_m, but it is ranked very high. On the right, we show the distribution of the publication timestamps of the documents in the lists to be combined. The vertical axis indicates the combined scores of posts with the same timestamps based on a baseline fusion method, e.g., CombSUM. According to its publication timestamp, d_2 was published in a "good" period for the query: many posts published around the same time as d_2 are highly ranked in many lists; because of this, BurstFuseX will "reward" d_2. In contrast, d_8 does not have a publication time around which many highly ranked posts were published, hence it should not receive a reward.

1994), CombMNZ (Lee, 1995), and one state-of-the-art supervised fusion methods: λ-Merge (Sheldon et al., 2011). For further comparisons, we consider a number of burst or time-sensitive microblog retrieval baselines. As BurstFuseX detects bursts based on the output of a standard fusion method rather than on the contents of microblogs, we also consider a baseline that detects bursts based on the contents. As we will see below, BurstFuseX significantly outperforms most fusion approaches and burst or time-sensitive retrieval methods.

Our contributions in this paper can be summarized as follows.

i. We propose a novel and effective probabilistic data fusion model to microblog post search, BurstFuseX, which not only takes traditional information such as document rank into account, but also exploits the temporal characteristics of microblog environments.
ii. To the best of our knowledge, this is the first attempt to frame the problem of searching microblog posts as a data fusion problem and also the first attempt to integrate temporal characteristics of result sets into data fusion.
iii. We provide a detailed analysis of the performance of BurstFuseX and offer a number of examples where we observe the effect hypothesized in Fig. 1, i.e., of posts in a burst having their rank boosted.

In Section 2 we discuss related work; in Section 3 we detail BurstFuseX; we follow with a description of our experimental setup in Section 4 and report on our experimental results and perform topic-level and run-time analyses in Section 5. Finally, Section 6 concludes the paper.

2. Related work

In this section, we first review data fusion approaches in the information retrieval literature, briefly discuss previous ways of performing burst detection in information retrieval, and finally we survey state-of-the-art approaches of searching microblogs since the TREC 2011 Microblog track ran.

2.1. Data fusion

The task of fusing document lists that have been retrieved in response to a given query so as to compile a single more effective result list has been widely studied in the information retrieval literature (Beitzel et al., 2003; Dong & Srivastava, 2013; He & Wu, 2008; Kozorovitsky & Kurland, 2011; Montague & Aslam, 2002; Shaw et al., 1994; Sheldon et al., 2011; Shokouhi & Si, 2011; Tsai et al., 2008; Wu, 2012). Data fusion have a large number of applications, e.g., in multilingual information retrieval (Sheldon et al., 2011; Si, Callan, Cetintas, & Yuan, 2008), federated search (He, Hong, & Si, 2011; Hong & Si, 2012; Shokouhi & Si, 2011) also known as distributed retrieval (Crestani & Markov, 2013), resource selection (Hong & Si, 2013; Markov, Azzopardi, & Crestani, 2013b; Markov & Crestani, 2014), etc. We divide these existing data fusion approaches into supervised and unsupervised methods.

Supervised data fusion approaches first extract a large number of features, either from documents or lists, and then utilize a machine learning algorithm to train the fusion model (Croft, 2000; Efron, 2011; Sheldon et al., 2011; Tsai et al., 2008; Wu, 2012). Supervised data fusion approaches become feasible when we can leverage the use of information existing in labeled training data. Adopting a supervised learning approach to data fusion has some advantages. For instance, we can apply existing optimization techniques to the data fusion problem, and the approaches become more easily amenable to specific domains or user groups (Liu, Liu, Qin, Ma, & Li, 2007). Liu et al. (2007) set up a general framework for conducting supervised data fusion, in which learning is formalized as an optimization problem in which one minimizes disagreements between ranking results and the labeled data. Tsai et al. (2008) propose a learning approach for the merging process in multilingual information retrieval. To conduct the learning data fusion approach, they extract a number of features from the given query, the documents to be retrieved and the translation, and then use an existing learning to rank algorithm to construct a merge model from a large number of labeled data. Qin, Geng, and Liu (2010) propose a supervised probabilistic data fusion model, which is based on coset-permutation distance and defined in a stage-wise manner. To fuse result lists generated by different query reformulations, the state-of-the-art data fusion method λ-Merge proposed by Sheldon et al. (2011) first extracts features from both the lists and the documents appearing in any of the lists, and then uses a learning to rank method to optimize a given metric, like NDCG, MAP, to fuse the lists into a final merging list in response to a given query. We use λ-Merge as a representative example of supervised fusion methods. Recently, Hong and Si (2012) propose a novel supervised fusion model for result merging by utilizing multiple centralized retrieval algorithms. However, the fact that a large amount of labeled data has to be available, together with other supervised problems (for instance over-fitting noted above in λ-Merge), makes supervised data fusion less useful when labeled data is hard to come by. Our experimental results show that in many cases, even a traditional unsupervised data fusion can beat a state-of-the-art supervised data fusion method.

In contrast, *unsupervised* data fusion methods mainly utilize either retrieval scores or ranks of documents in the lists to be merged (Bruno & Marchand-Maillet, 2009; Croft, 2000; Khalaman & Kurland, 2012; Shaw et al., 1994; Wu, 2012). Methods utilizing retrieval scores take score information from the result lists to be fused as input, while those utilizing rank information only use order information of the documents appearing in any of the lists to be fused as input. Data fusion methods

utilizing rank information have many uses and applications in information retrieval, including, e.g., meta-search (Aslam & Montague, 2001; Shaw et al., 1994) where only order information from the result lists tends to be available. Our burst-aware data fusion algorithm only uses rank and time information of the posts in the result lists, which makes it usable in cases where only order information is available.

Unsupervised data fusion has a long history with the CombSUM family of fusion methods being the oldest and one of the most successful ones in many information retrieval tasks (Croft, 2000; Kozorovitsky & Kurland, 2011; Shaw et al., 1994; Tsagkias et al., 2011). Other unsupervised data fusion approaches include, for instance, Borda Count (Aslam & Montague, 2001; Dwork et al., 2001; Erp & Schomaker, 2000), median data fusion (Fagin, Kumar, & Sivakumar, 2003), genetic algorithm (Beg, 2004), fuzzy logic based data fusion (Ahmad & Beg, 2002), Markov Chain based data fusion (Dwork et al., 2001), the outranking model for fusion (Farah & Vanderpooten, 2007), data fusion in clustering microarray data (Kustra & Zagdanski, 2010), data fusion for the management of multimedia documents (Deloule, Lambert, Beauchene, & Ionescu, 2007) and a distance-based model (Klementiev, Roth, & Small, 2008). In addition, Markov and Crestani (2014) and Markov, Arampatzis, and Crestani (2012, 2013a) provide theoretical arguments on why some traditional unsupervised fusion methods work, and based on these insights, they propose other unsupervised fusion methods. Through the use of unsupervised data fusion, Loia, Pedrycz, and Senatore (2007) offer a new way of organizing web documents that emphasizes a direct separation between syntactic and semantic facets.

Khalaman and Kurland (2012) utilize the content of documents appearing in any of the result lists to be fused to get an additional source of rich information, i.e., document similarities, and then integrate information induced from the clusters of similar documents created across the result lists to be merged with the output of a fusion method that relies on retrieval scores. This fusion model makes strong assumptions: the content of documents is assumed available and it is very easy to compute document similarities and get clusters for documents. However, in the case of microblog retrieval, some of these assumptions are somewhat unrealistic. For instance, some posts with only links but without any words are still labeled as relevant ones in response to the query, and creating clusters of similar posts may be very challenging as the length of posts is at most 140 characters, while many posts are ambiguous (Liang, Ren, & de Rijke, 2014a, 2014b; Zhao & Rosson, 2009). Instead of computing document similarities and creating clusters, we make full use of the characteristics of microblog environments where people tend to talk about a specific topic within specific time intervals. Our data fusion method utilizes the timestamp of posts. We detect bursts—sets of documents that are generated in specific time windows—, and use posts within a burst to boost the scores of "nearby" posts. To our knowledge, this is the first attempt to integrate the temporal characteristics of posts into data fusion.

2.2. Burst detection

Our framework for detecting bursts is similar to that in (Chen et al., 2010; Hoonlor et al., 2012; Lappas et al., 2009; Mathioudakis et al., 2010; Peetz et al., 2012; Vlachos et al., 2004; Zhu & Shasha, 2003), but the input information we use and our purpose in detecting bursts differs strongly. Much prior work detects bursts mainly based on document frequency or/and query term frequency in the whole corpus (Hoonlor et al., 2012; Peetz et al., 2012; Zhu & Shasha, 2003). In contrast, the information in our burst detection method is the score of documents generated by standard data fusion algorithm from documents in the lists to be merged. Our purpose in detecting bursts is also different to the aims in (Chen et al., 2010; Hoonlor et al., 2012; Lappas et al., 2009; Peetz et al., 2012; Vlachos et al., 2004). For instance, work in (Chen et al., 2010) detects bursts for generating events and their evolution in new streams, while we detect bursts to help improve the effectiveness of fusing methods. Unlike most of the past work on detecting bursts (Chen et al., 2010; Hoonlor et al., 2012; Lappas et al., 2009; Miyanishi, Seki, & Uehara, 2013a; Peetz et al., 2012; Vlachos et al., 2004), we do not make any assumption that the content of documents is available and the distribution of documents is known. In contrast, we only utilize the rank information of posts appearing in the result lists to be fused to detect bursts, and our experimental results show that using standard fusion scores to detect bursts outperforms using the content of posts to detects bursts.

2.3. Microblog retrieval

Microblog retrieval has become an active research topic in IR, especially following the launch of the Microblog track at TREC in 2011 (Ounis et al., 2011). Earlier work, however, already explored the task of retrieving microblog posts. O'Connor, Krieger, and Ahn (2010) present *TweetMotif*, an exploratory search application for Twitter. Unlike traditional approaches to information retrieval, which present a simple list of messages, *TweetMotif* groups messages by frequent significant terms, a result set's subtopics, which facilitate navigation and drilldown through a faceted microblog search interface. Efron (2010) proposes a language model for hashtag retrieval in a microblog environment, where retrieved hashtags on a topic of interest for query expansion are utilized to improve the performance of microblog search. Duan, Jiang, Qin, Zhou, and Shum (2010) show that learning to rank methods work well on the task of microblog retrieval and that account authority and URL presence are very strong features.

Following the launch of the Microblog track at TREC in 2011, many approaches have been proposed. At TREC in 2011, for instance, some approaches (Amati et al., 2011; Cao, Gao, Yu, Liu, & Cheng, 2011; Horn, Pimas, Granitzer, & Lex, 2011;

Metzler & Cai, 2011; Wei, Gao, Zhou, Li, & Wong, 2011) exploit the idea that microblog queries are distinguished from web queries with many unique characteristics, and utilize the temporal information to help searching posts. The method proposed by Metzler and Cai (2011) combines a Markov random field model with a learning to rank model for searching posts, which achieves the best p@30 performance at TREC in 2011. A combination strategy is also used by Zhang, Hui, He, and Luo (2011) to search posts in 2011, where they combine a field-based model that takes the frequency of a query term in different document fields into account with query expansion. In contrast, work present in (Bandyopadhyay, Mitra, & Majumder, 2011) uses query expansion only for searching posts, but the way their query expansion method works is different; they use the Google Search API to retrieve pages from the web, and use the titles to expand the queries.

At the TREC 2012 Microblog track, Luo, Osborne, Petrovic, and Wang (2012) consider a microblog post to be a structured document, consisting not only of the text, but also of other blocks, like hashtags, links, and mentions. Using these blocks as features in a learning to rank method, they show good retrieval performances. At TREC 2012, the best performing run also uses learning to rank model (Han et al., 2012). Wei, Zhang, Li, and Wang (2012) propose a ranking algorithm with temporal information based on a language model. Kim, Yeniterzi, and Callan (2012) present two approaches to address the problem of the limited vocabulary of each posts due to their short length. The first is query expansion through pseudo-relevance feedback and the other is document expansion of tweets using web documents linked from the body of a tweet. Jabeur et al. (2012) experiment with a bayesian network retrieval model for posts search and a feature learning model for relevance classification.

Beside the approaches presented at TREC 2011 and 2012, many microblog post retrieval approaches have been proposed outside TREC since the launch of the TREC 2011 Microblog track. For instance, Massoudi, Tsagkias, de Rijke, and Weerkamp (2011) and Miyanishi et al. (2013a) propose a method for query expansion in the microblog domain and find that this is highly effective. Naveed, Gottron, Kunegis, and Che Alhadi (2011) explore the impact of document length normalization on retrieval performance and find that this has a negative effect. They also introduce *interestingness* as a measure for microblog posts and show that using this measure leads to better retrieval effectiveness. Choi, Croft, and Kim (2012) suggest a quality model using surrogate judgments based on retweets that can be collected automatically to train a microblog search model. Chang et al. (2013) propose a method to utilize Twitter TinyURL (shortened URL links) to detect fresh and high-quality documents, and leverage Twitter data to generate novel and effective features for ranking documents. The work by Miyanishi, Seki, and Uehara (2013b), Dakka, Gravano, and Ipeirotis (2012), Choi and Croft (2012) and Massoudi et al. (2011) utilizes burst (time) information to boost the performance of searching posts.

Another related line of work concerns retrieval score regularization for improving the performance of ad hoc search (Diaz, 2005, 2007). Specifically, Diaz (2005, 2007) present a framework for improving document retrieval scores under a regularization framework, where retrieval scores of documents are adjusted to respect inter-document consistency. Our microblog search algorithm, BurstFuseX, builds on the same intuitions as the algorithms proposed in (Diaz, 2005, 2007): in (Diaz, 2005, 2007) the regularization algorithms for ad hoc search build on the cluster hypothesis, according to which closely related documents should have similar retrieval scores, given the same information request; in our algorithm, we assume that posts published within the same time intervals are more likely to be talking about the same topic. There are several dissimilarities, though, between the algorithms in (Diaz, 2005, 2007) and our algorithm. For instance, the algorithms in (Diaz, 2005, 2007) utilize (expensive) inter-document similarities for retrieval score regularization; in contrast, our algorithm utilizes time information for data fusion score regularization. The input of the former algorithms consists of the retrieval scores of documents generated by a single retrieval model and the documents' content; in contrast, the input of our algorithm is a number of result lists generated by multiple retrieval models plus the timestamps of the posts.

In sum, the work that we present in this paper differs in important ways from the related work just discussed. We observe that people tend to talk about topics within specific time intervals. We first detect set of bursts where the majority of posts appear in many of the lists and are likely to be relevant. Then we let posts published in or near the burst boost each others' scores. To our knowledge, this is the attempt to frame the problem of searching microblog posts as a data fusion problem and the first attempt to integrate temporal characteristics of result sets into data fusion techniques.

3. Fusion approach

In this section, we first provide the main research question we address. Then we briefly describe standard unsupervised and supervised data fusion methods that will be integrated in our proposed data fusion methods and taken as baselines in our experiments. We define what bursts are and detail how to detect bursts in data fusion scenarios in Section 3.2. After that, we detail our proposed data fusion methods for microblog search in Section 3.3.

The task we address in this paper is the following: *Given a query and a set of ranked lists of posts returned in response to the query, fuse the lists into a single ranked list of posts to be returned in response to the query*. Hence, the input of our burst-aware data fusion method BurstFuseX consists of a query and a set of ranked lists of posts; the output is a single fused list. Algorithm 1 gives a high level overview of BurstFuseX.

Algorithm 1. BurstFuseX: Burst-aware data fusion for microblog post search

Input: A query q
A number of ranked lists of posts to be fused, $L_1, L_2, \ldots, L_m$
The combined set of posts $\mathcal{C}_{\mathcal{L}} := \bigcup_{i=1}^{m} L_i$
A standard fusion method X.
Output: A final fused list of posts.
1 Calculate the (standard) fusion score $F_X(d;q)$ according to X for each post $d \in \mathcal{C}_{\mathcal{L}}$; see Section 3.1;
2 Detect bursts based on the timestamps and $F_X(d;q)$ scores; see Section 3.2;
3 Calculate the BurstFuseX fusion score for each $d \in \mathcal{C}_{\mathcal{L}}$ using the bursts and the standard fusion score; see Section 3.3;
4 Construct the final fused list based on the BurstFuseX score of $d \in \mathcal{C}_{\mathcal{L}}$ obtained in step 3.

In the remainder of this section we detail the steps that make up BurstFuseX. In Table 1 we list the notation that we use.

The fusion methods we consider as building blocks for BurstFuseX all assign a non-negative *fusion score* $F_X(d;q)$ to every post $d \in \mathcal{C}_{\mathcal{L}}$. We set $F_X(d;q) := 0$ for $d \notin \mathcal{C}_{\mathcal{L}}$, following (Bruno & Marchand-Maillet, 2009; Kozorovitsky & Kurland, 2011; Lee, 1995; Shaw et al., 1994; Wu, 2012). The higher $F_X(d;q)$ is, the more likely d is assumed to be an appropriate response to q.

3.1. Standard fusion methods

To be able to define the final fusion score $F_{BurstFuseX}(d;q)$ we integrate and make use of an existing standard fusion method (step 1 of Algorithm 1). BurstFuseX is independent of the particular choice of the standard fusion method that it integrates: any fusion method can be integrated into our model. Below, we briefly review the three standard fusion methods that we consider in this paper: two unsupervised ones and a supervised method.

3.1.1. Unsupervised fusion

Classical unsupervised methods include the so-called CombSUM family. Methods in this family assume that documents that are ranked highly in many of the lists to be merged are highly relevant (Shaw et al., 1994): typically, a fusion score $F_X(d;q)$ for document d, given query q, is defined based on the rank of d in the lists to be merged and on the number of lists in which d appears.

Let $R_{L_i}(d)$ denote d's score based on the rank of d in list L_i; by default, $R_{L_i(d)} = 0$ if $d \notin L_i$. In both CombSUM and CombMNZ, $R_{L_i}(d)$ is often defined as:

Table 1
Notation used in the paper.

Notation	Gloss
$\mathcal{C}$	Corpus of microblog posts
q	Query
d	Microblog post
L_i	Ranked list of microblog posts
$\mathcal{L}$	Set of ranked lists
$\mathcal{C}_{\mathcal{L}}$	Set of posts that appear in the lists in $\mathcal{L}$
X	Standard fusion method
$F_X(d;q)$	Score of post d for query q according to standard fusion method X
$R_{L_i}(d)$	Rank-based score of d in list L_i
$\text{rank}(d, L_i)$	Rank of d in list L_i
k_{L_i}	Length of list L_i
α_m	Weight of a list; used in the definition of λ-Merge
$g(d;q)$	Scoring function used in the definition of λ-Merge
$f(x;\theta)$	Linear scoring function used in the definition of λ-Merge
t_i	Timestamp
d_{t_i}	Post with timestamp t_i
$\mathcal{S}_{t_i}(\mathcal{C}_{\mathcal{L}})$	Burst-time score at time t_i
$t_{\mathcal{C}_{\mathcal{L}}}$	Number of different timestamps of posts in $\mathcal{C}_{\mathcal{L}}$
$\mathfrak{I}(\mathcal{C}_{\mathcal{L}})$	Sequence of burst-time scores
$b(\mathcal{C}_{\mathcal{L}})[t_i : t_j]$	Burst with start timestamp t_i and end timestamp t_j (given query q), abbreviated by b
$\mathcal{B}(\mathcal{C}_{\mathcal{L}})$	Set of all bursts in $\mathcal{C}_{\mathcal{L}}$ (given query q)
μ	Free parameter that governs burst information
σ_b	Standard deviation of timestamps belonging to the burst b

$$R_{L_i}(d) = \frac{(1 + k_{L_i}) - \text{rank}(d, L_i)}{k_{L_i}}, \tag{1}$$

where $\text{rank}(d, L_i) \in \{1, \ldots, k_{L_i}\}$ is the rank of d in L_i. The well-known CombSUM fusion method (Shaw et al., 1994; Wu, 2012), for instance, scores d by the sum of its rank scores in the lists:

$$F_{CombSUM}(d; q) := \sum_{L_i} R_{L_i}(d),$$

while CombMNZ (Shaw et al., 1994; Wu, 2012) rewards documents d that rank high in many lists:

$$F_{CombMNZ}(d; q) := |\{L_i : d \in L_i\}| \cdot F_{CombSUM}(d; q).$$

3.1.2. Supervised fusion

Recently, several supervised methods for merging ranked lists have been proposed, one of which is λ-Merge (Sheldon et al., 2011). In this paper, we view λ-Merge as a typical representative of the supervised standard fusion methods that are currently available.[2]

Given a query, λ-Merge can directly optimize a retrieval metric (e.g., MAP) to enhance retrieval effectiveness under the assumption that query reformulation candidates are available. In particular, λ-Merge learns a scoring function to rank documents from multiple reformulations of the given query by combining features that indicate document quality (such as retrieval score) with features that indicate the quality of the reformulation and its results lists (called *gating features* in Sheldon et al., 2011). In our setting, we do not assume that query reformulation candidates are easily available, i.e., no features about the quality of the reformulation (*gating features*) are used in our data fusion method.

Our settings for λ-Merge are detailed in an appendix to the paper (see Appendix A).

3.2. Bursts and burst detection

To ground our intuitions about utilizing burst information to boost the performance of microblog search, we choose four test queries as examples and examine plots of the number of relevant documents distributed over their document ages (measured by days) in Fig. 2.[3] The figure confirms that people tend to talk about topics within specific time windows. It is, therefore, worthwhile to detect such time windows ("bursts") and to use such burst information, which is what our proposed data fusion method for microblog search aims to do.

Next, we move on to the next step (step 2) of Algorithm 1 and detail how we detect bursts. Let t_i be a timestamp. Let d_{t_i} ($\in \mathcal{C}_\mathcal{L}$) denote a post d with timestamp t_i. We regard posts published during the same hour as having the same timestamp. Although it is possible to define "the same timestamp" in many different ways, we found that this is a suitable level of granularity for the fusion effectiveness of searching posts; the same setting is also used in (Metzler, Cai, & Hovy, 2012). Now, before we detect bursts, we need to define $\mathcal{S}_{t_i}(\mathcal{C}_\mathcal{L})$, the *burst-time score at time t_i of $\mathcal{C}_\mathcal{L}$*, the set of posts occurring in the lists under consideration. Let $F_X(d_{t_i}; q)$ be the score of d_{t_i} given q under the standard fusion method X. Then:

$$\mathcal{S}_{t_i}(\mathcal{C}_\mathcal{L}) = \frac{\sum_{d_{t_i} \in \mathcal{C}_\mathcal{L}} F_X(d_{t_i}; q)}{\sum_{j=1}^{t_{\mathcal{C}_\mathcal{L}}} \sum_{d_{t_j} \in \mathcal{C}_\mathcal{L}} F_X(d_{t_j}; q)} - \frac{1}{t_{\mathcal{C}_\mathcal{L}}}, \quad 1 \leqslant i \leqslant t_{\mathcal{C}_\mathcal{L}}, \tag{2}$$

where $1 \leqslant j \leqslant t_{\mathcal{C}_\mathcal{L}}$ and $t_{\mathcal{C}_\mathcal{L}}$ is the total number of different timestamps belonging to posts in $\mathcal{C}_\mathcal{L}$. Notice that the burst-time score $\mathcal{S}_{t_i}(\mathcal{C}_\mathcal{L}) > 0$ if it is above the average score (i.e., $1/t_{\mathcal{C}_\mathcal{L}}$), otherwise $\mathcal{S}_{t_i}(\mathcal{C}_\mathcal{L}) \leqslant 0$.

We compute a burst-time score $\mathcal{S}_{t_i}(\mathcal{C}_\mathcal{L})$ at each time point $t_i \in \{t_1, t_2, \ldots, t_{\mathcal{C}_\mathcal{L}}\}$ in $\mathcal{C}_\mathcal{L}$. In this manner we generate a *burst-time score sequence* $\mathfrak{I}(\mathcal{C}_\mathcal{L}) = \{\mathcal{S}_{t_1}(\mathcal{C}_\mathcal{L}), \mathcal{S}_{t_2}(\mathcal{C}_\mathcal{L}), \ldots, \mathcal{S}_{t_{\mathcal{C}_\mathcal{L}}}(\mathcal{C}_\mathcal{L})\}$.

Following (Ruzzo & Tompa, 1999), a segment $\mathfrak{I}(\mathcal{C}_\mathcal{L})[t_i : t_j] = \{\mathcal{S}_{t_i}(\mathcal{C}_\mathcal{L}), \mathcal{S}_{t_{i+1}}(\mathcal{C}_\mathcal{L}), \ldots, \mathcal{S}_{t_j}(\mathcal{C}_\mathcal{L})\}$, where $1 \leqslant i \leqslant j \leqslant t_{\mathcal{C}_\mathcal{L}}$, is a *maximal segment* in $\mathfrak{I}(\mathcal{C}_\mathcal{L})$ if:

i. All proper subsequences of $\mathfrak{I}(\mathcal{C}_\mathcal{L})[t_i : t_j]$ have a lower score.[4]
ii. No proper super-segments of $\mathfrak{I}(\mathcal{C}_\mathcal{L})[t_i : t_j]$ in $\mathfrak{I}(\mathcal{C}_\mathcal{L})$ satisfy item i.

We adapt a linear-time algorithm proposed in (Ruzzo & Tompa, 1999) to find *all maximal segments* in the sequence $\mathfrak{I}(\mathcal{C}_\mathcal{L})$. As an example, consider the input sequence $\mathfrak{I}(\mathcal{C}_\mathcal{L}) = \{2, -2, 4, 3, -3, -4, -1, -3, 5, -1, 3, -2\}$. The maximal segments in this sequence are $\{2\}$, $\{4, 3\}$ and $\{5, -1, 3\}$. The segment $\{2, -2, 4, 3\}$ is not maximal, since it has a nonempty zero-scoring prefix $\{2, -2\}$ appending to the left of $\{4, 3\}$; $\{5\}$ is not a maximal segment, since $\{5, -1, 3\}$ has a total higher score of 7. Each maximal segment $\mathfrak{I}(\mathcal{C}_\mathcal{L})[t_i : t_j]$ gives rise to a *burst* of posts $b(\mathcal{C}_\mathcal{L})[t_i : t_j]$ with start timestamp t_i and end timestamp t_j: it contains

[2] To be able to define λ-Merge, we need to assume that we can access the content of posts.
[3] The topics are selected from test collections detailed in Section 4.2.
[4] The score of a subsequence is the sum of the burst-time scores of the elements in the subsequence.

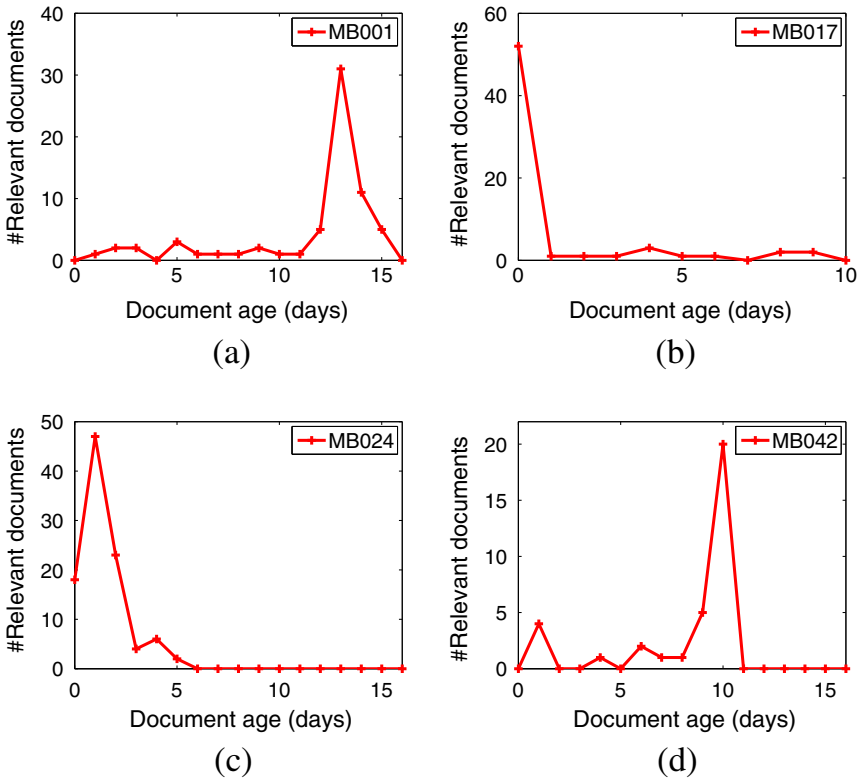

Fig. 2. Distribution of the number of relevant documents over days for four test queries. In each subfigure, the *x*-axis indicates document ages from query time to the document timestamps, and the *y*-axis indicates the number of relevant documents according to the ground-truth in the Tweets 2011 dataset (detailed in Section 4.2). Subfigure (a) plots the relevant documents over time for query MB001 – *BBC World Service staff cuts*, (b) is for MB017 – *White Stripes breakup*, (c) is for MB024 – *Super Bowl, seats*, and (d) is for MB042 – *Holland Iran envoy recall*.

any post $d \in \mathcal{C}_{\mathcal{L}}$ whose timestamp is between t_i and t_j is within this segment. We write $\mathcal{B}(\mathcal{C}_{\mathcal{L}}) = \bigcup b(\mathcal{C}_{\mathcal{L}})[t_i : t_j]$ to denote the *set of all bursts in response to q*.

We let b be short for $b(\mathcal{C}_{\mathcal{L}})[t_i : t_j]$ in the following. As it does not access the contents of posts, the source of complexity in our burst detection method is in the problem of finding all maximal segments: this problem can be solved in linear time (Ruzzo & Tompa, 1999), so that the computational complexity of our burst detection method is $O(|\mathcal{C}_{\mathcal{L}}|)$.

3.3. Burst-aware fusion

We turn to the key steps 3 and 4 of Algorithm 1 and define our burst-aware fusion algorithm. Motivated by the fact that people tend to talk about a topic within specific short time intervals (Lappas et al., 2009; Mathioudakis et al., 2010; Peetz et al., 2012), we devise a method that allows posts in the same burst to boost the scores of other posts such that posts that are ranked low in a small number of lists can be promoted by posts in the same burst that are ranked highly in many lists. After detecting a set of bursts, $\mathcal{B}(\mathcal{C}_{\mathcal{L}})$, we integrate burst information with scores of posts in the lists, scores that were generated by a standard fusion method X, to estimate $P(d|q)$—the final probability that d ($\in \mathcal{C}_{\mathcal{L}}$) is relevant to query q.

3.3.1. The model

We use a set of bursts $\mathcal{B}(\mathcal{C}_{\mathcal{L}})$ as proxies for d in estimating its relevance in response to q. Specifically, we can rewrite the probability of a post d being about $q, P(d|q)$, as:

$$P(d|q) = \sum_{b \in \mathcal{B}(\mathcal{C}_{\mathcal{L}})} p(d|b,q) \cdot p(b|q), \tag{3}$$

where the probability, $p(b|q)$, indicates how likely a set of posts in b produced within the same time interval are relevant to q, and $p(d|b,q)$ indicates how likely d is talking about q and belongs to b. To estimate $p(d|b,q)$, a linear mixture governed by a free parameter μ is used (Kurland & Lee, 2004; Markovits, Shtok, & Kurland, 2012) such that:

$$p_{\theta}(d|b,q) \;:=\; (1-\mu) \cdot p(d|q) + \mu \cdot p(d|b), \tag{4}$$

where $p(d|q)$ measures the relevance of d to q and $p(d|b)$ indicates how likely d belongs to b. We substitute Eq. (4) into Eq. (3), and define our *BurstFuseX* model as:

$$F_{BurstFuseX}(d;q) := \sum_{b\in\mathcal{B}(\mathcal{C}_\mathcal{L})} \{(1-\mu)\cdot p(d|q) + \mu\cdot p(d|b)\}\cdot p(b|q) = (1-\mu)\cdot p(d|q) + \mu\cdot \sum_{b\in\mathcal{B}(\mathcal{C}_\mathcal{L})} p(d|b)\cdot p(b|q). \quad (5)$$

That is, to obtain a score for post d in response to q, BurstFuseX uses three types of probability. If posts in a burst are talking about q, b will be rewarded as $p(b|q)$ indicates. If d is strongly associated with b, then as indicated by $p(d|b), d$ will be rewarded. Finally, if each burst b in $\mathcal{B}(\mathcal{C}_\mathcal{L})$ talks about q, d itself discusses q and is strongly associated with a burst, then d will be ranked high in the final fused list.

Notice how BurstFuseX can boost the score of posts: if post d ranks low in a single list (i.e., $p(d|q)$ is small) but is contained in a burst, as indicated by $p(d|b)$, then the final fused score of d, $F_{BurstFuseX}(d;q)$, may still be relatively high, which may boost the final ranking of d.

3.3.2. Estimating the key components

Our next task is to derive estimates for the following key components in Eq. (5):

- $p(d|q)$: post-level relevance—how likely d is talking about q.
- $p(b|q)$: burst-level relevance—how likely a set of posts as a whole are talking about q.
- $p(d|b)$: post-burst association strength—how likely d belongs to b.

Post-level relevance. To obtain $p(d|q)$ in Eq. (5), we apply Bayes' Theorem, such that $p(d|q) = \frac{p(q|d)p(d)}{p(q)}$, where we let $p(q) \propto \sum_{d'\in\mathcal{C}_\mathcal{L}} p(q|d')p(d')$. Here, d' is a post in the set of posts $\mathcal{C}_\mathcal{L}$ to be fused. A uniform prior distribution is assumed for each post $d' \in \mathcal{C}_\mathcal{L}$. So $p(d|q)$ can be rewritten as:

$$p(d|q) \propto \frac{p(q|d)}{\sum_{d'\in\mathcal{C}_\mathcal{L}} p(q|d')}.$$

We use an estimate $p_\theta(q|d) \propto F_X(d;q)$ (Khalaman & Kurland, 2012), where $F_X(d;q)$ is the score of a standard fusion method X for d given q:

$$p_\theta(d|q) := \frac{F_X(d;q)}{\sum_{d'\in\mathcal{C}_\mathcal{L}} F_X(d';q)}, \quad (6)$$

which is the normalized standard fusion score reflecting d's relevance to q. Notice that our burst-aware fusion model will reduce to the standard fusion method X if we let $\mu = 0$ in Eq. (5), as $F_{BurstFuseX}(d;q) = (1-\mu)\cdot p(d|q) \propto F_X(d|q)$ in Eq. (5) in this case. In other words, the effect of merging result lists according to $F_{BurstFuseX}(d;q)$ will then be the same as that of merging result lists according to $F_X(d;q)$.

Burst-level relevance. Next, to obtain $p(b|q)$ in Eq. (5), we apply Bayes' Theorem again, such that $p(b|q) = \frac{p(q|b)p(b)}{p(q)}$, where we use the probability rule, and have $p(q) \propto \sum_{b'\in\mathcal{B}(\mathcal{C}_\mathcal{L})} p(q|b')p(b')$. Here, b' is a burst in the set of bursts $\mathcal{B}$ detected by our burst detection method. Assuming a uniform prior for each burst in $\mathcal{C}_\mathcal{L}$ for a given q, the probability that a burst b contains information pertaining to q can be represented as:

$$p(b|q) \propto \frac{p(q|b)}{\sum_{b'\in\mathcal{B}(\mathcal{C}_\mathcal{L})} p(q|b')}.$$

A burst may contain posts that have multiple appearances in the lists to be fused. Prior work on representing sets of posts has shown that product-based representations somewhat outperform sum-based representations (Khalaman & Kurland, 2012; Liu & Croft, 2008; Seo & Croft, 2010). Accordingly, we let:

$$p_\theta(q|b) = \prod_{d\in b} p(q|d)^{\frac{1}{|b|}}.$$

As we use an estimate $p_\theta(q|d) \propto F_X(d;q)$ (see above), $p(b|q)$ can be estimated as:

$$p_\theta(b|q) := \frac{\prod_{d\in b} F_X(d;q)^{\frac{1}{|b|}}}{\sum_{b'\in\mathcal{B}(\mathcal{C}_\mathcal{L})} \prod_{d'\in b'} F_X(d';q)^{\frac{1}{|b'|}}}, \quad (7)$$

where $|b|$ and $|b'|$ are the number of posts in b and b', respectively.

Post-burst association strength. To obtain $p(d|b)$ in Eq. (5), we apply Bayes' Theorem again, such that $p(d|b) = \frac{p(b|d)p(d)}{p(b)}$. We observe that $p(b) \propto \sum_{d'\in\mathcal{C}_\mathcal{L}} p(b|d')p(d')$ and assume a uniform prior for the probability of a post, so that $p(d|b)$ can be represented as:

$$p(d|b) \propto \frac{p(b|d)}{\sum_{d'\in\mathcal{C}_\mathcal{L}} p(b|d')}.$$

Here, $p(b|d)$ is the probability of d belonging to b.

Next, we need to estimate $p(b|d)$. Again, we use the product of scores of posts in a burst with the index of $1/|b|$ rather than the average of the sum of the score.[5] We set:

$$p_\theta(b|d) := \prod_{d'' \in b} p(d''|d)^{\frac{1}{|b|}}, \tag{8}$$

to estimate $p(b|d)$, where $d'' \in \mathcal{C}_\mathcal{L}$ is a post in b.

Three factors affect the association strength between d and b: the temporal relationship between d and posts $d'' \in b$, the relevance of d given q and the relevance of d'' given q. We estimate the time relationship between d and d'' as:

$$p_t(d'', d) = \exp\left\{-\frac{(t_{d''} - t_d)^2}{2\sigma_b^2}\right\}.$$

Here, t_d and $t_{d''}$ are the timestamps of post d and d'', respectively, σ_b is the standard deviation of the timestamps in b:

$$\begin{aligned}\sigma_b^2 &= \frac{\sum_{k=1}^{n_b}\left\{k - \frac{n_b+1}{2}\right\}^2}{n_b} = \frac{\sum_{k=1}^{n_b}\left\{k^2 - k(n_b+1) + \frac{n_b^2+2n_b+1}{4}\right\}}{n_b} = \frac{\frac{n_b^3+2n_b^2+n_b}{4} + \sum_{k=1}^{n_b}k^2 - (n_b+1)\sum_{k=1}^{n_b}k}{n_b} \\ &= \frac{\frac{n_b^3+2n_b^2+n_b}{4} + \frac{n_b(n_b+1)(2n_b+1)}{6} - (n_b+1)\frac{(n_b+1)n_b}{2}}{n_b} = \frac{n_b^2-1}{12},\end{aligned} \tag{9}$$

where $n_b = j - i + 1$ is the total number of different timestamps of posts in the burst b.[6] If $j = i$, we let $\sigma_b = 0.5$ to avoid $\sigma_b = 0$. The bigger the temporal distance between t_d and $t_{d''}$ is, the smaller $p_t(d'', d)$ will be, which means that compared to other posts in burst b, d is rewarded less by post d''.

Now, to estimate $p(d''|d)$ (Eq. (8)) we build on the following intuition. If d'' is ranked highly, based on a relatively large value of $p(d''|q)$, and d'' and d are produced at almost the same points in time, based on a relatively high value of $p_t(d'', d)$, then d'' should be able to boost d's score. Hence, we estimate $p(d''|d)$ by putting $p_\theta(d''|d) := p(d''|q) \cdot p_t(d'', d) := p_\theta(d''|q) \cdot p_t(d'', d)$. When we substitute this term in (8) we obtain:

$$p_\theta(b|d) := \prod_{d'' \in b} \{p_\theta(d''|q) \cdot p_t(d'', d)\}^{\frac{1}{|b|}}.$$

Putting everything together, we can now estimate the post-burst association strength, $p(d|b)$, as:

$$p_\theta(d|b) = \frac{\prod_{d'' \in b}\{p_\theta(d''|q) \cdot p_t(d'', d)\}^{\frac{1}{|b|}}}{\sum_{d' \in \mathcal{C}_\mathcal{L}}\prod_{d'' \in b}\{p_\theta(d''|q) \cdot p_t(d'', d')\}^{\frac{1}{|b|}}}. \tag{10}$$

According to (10), if d is in b and the scores of posts surrounding d (including d itself) in b are high, the association strength between d and b increases. In this case, d's scores will be boosted. Note, by the way, that d ($\in \mathcal{C}_\mathcal{L}$) does not have to be in b; any d in $\mathcal{C}_\mathcal{L}$ can have a non-zero association strength to any b in $\mathcal{B}(\mathcal{C}_\mathcal{L})$.

4. Experimental setup

In this section, we describe our experimental setup; Section 4.1 lists our specific research questions; Section 4.2 describes the data set; Section 4.4 details the evaluation metrics and the significance testing used in the experiments. Finally, Sections 4.5 and 4.6 detail how BurstFuseX is trained and optimized, and the settings of the experiments.

4.1. Research questions

The research questions guiding the remainder of the paper are:

- i Does BurstFuseX outperform the standard data fusion method that it integrates? (See Section 5.1).
- ii Does BurstFuseλ-Merge outperform BurstFuseCombSUM or BurstFuseCombMNZ? (See Section 5.1).
- iii Does BurstFuseX outperform the best run to be fused? (See Section 5.1).
- iv What is the effect of using burst information in BurstFuseX? I.e., what is the impact of the free parameter μ in Eq. (5)? (See Section 5.2).
- v What is the effect of the number of lists to be fused in BurstFuseX? (See Section 5.3).
- vi Can we observe the hypothesized effect sketched in Fig. 1 (See Section 5.4).
- vii How fast is BurstFuseX compared to other data fusion methods? (See Section 5.5).

[5] Experimental results show that using products is not statistically significantly different from using sums with a two-tailed paired t-test at a 95% confidence level.

[6] Alternative definitions of σ_b are possible, but we found that this has little effect on the overall retrieval performance.

Table 2
Description of the data set used in our experiments.

Number of tweets	15,137,399
Number of users	4,670,516
Median tweet length	8.66
Median English tweet length	10.76
Number of English tweets	9,318,772
Number of English retweets	1,069,006
Number of hyperlinks	1,135,720
Number of hashtags	1,005,343

viii Can BurstFuseX beat burst or time-sensitive microblog search algorithms? (See Sections 5.6 and 5.7).
ix Can BurstFuseX aid a single run that does not take time into account? (See Section 5.8).

4.2. Data set

In order to answer our research questions we work with the Tweets 2011 corpus (Macdonald, Ounis, Lin, Choudhury, & Soboroff, 2011), called Tweet11, provided by the TREC 2011 Microblog track. The collection is comprised of approximately 16 million tweets collected over a period of 2 weeks (23th January until 8th February 2011, inclusive) sampled courtesy of Twitter. Different types of tweets in this data set are present, including replies and retweets. Each tweet has its own timestamp. Descriptive statistics about the collection are provided in Table 2.

The task studied at the TREC 2011 Microblog track was: given a query with a timestamp, return relevant and interesting tweets in reverse chronological order. This task is akin to adhoc search on Twitter, where a user's information need is represented by a query at a specific time. For 2012, the setting of the TREC was almost the same as that in 2011 except that the topics were different and the result lists were required to be ordered by relevance instead of chronologically (Soboroff et al., 2012). In our experiments, we rank tweets by relevance.[7]

We use two sets of test topics (queries) in our experiments, the 2011 test set and the 2012 test set. In total, NIST (the National Institute of Standards and Technology) created 50 test topics for TREC 2011 Microblog track, each representing an information need at a specific point in time when the topics were issued. Fig. 3 shows an example topic. 49 test topics were used in the TREC and 2965 tweets were deemed relevant; some topics have just two relevant tweets while some have more than 100 relevant tweets. Indeed, one of the 50 topics originally created, MB050, did not have any relevant tweets in the pool, and it was therefore dropped from the evaluation. To assess the tweets, the assessors judged the relevance of a tweet after reading it. Tweets in the Tweet11 corpus were judged on the basis of the defined information need using a three-point scale: *Not Relevant*, *Minimally Relevant* and *Highly Relevant* (Macdonald et al., 2011).

A total of 59 groups participated in the TREC 2011 Microblog track, with each team submitting at most four runs, which resulted in 184 runs[8] (Macdonald et al., 2011; Ounis et al., 2011). The official evaluation metric was precision at 30 (p@30) (Macdonald et al., 2011). The p@30 scores of these 184 runs varied dramatically, with the best run achieving a p@30 score of 0.4551 and the worst run achieving 0.000. In our experiments below, we do not use any runs whose p@30 scores are below 0.10, leaving us with 174 runs from the TREC 2011 Microblog track. Details about the implementation of each run from the TREC 2011 Microblog track can be found in (Macdonald et al., 2011; Ounis et al., 2011).

The Microblog search track continued in 2012 using the same corpus, Tweet11 (Soboroff et al., 2012). NIST created 60 new test topics representing information needs at specific points in time in TREC 2012 and labeled 6286 tweets as minimally or highly relevant. The TREC 2012 Microblog track received 121 runs[8] from 33 participating groups. The best run in TREC 2012 Microblog track is hitURLrun3 (Han et al., 2012), with its p@30 score being 0.4695. Once again, in our experiments, we only use the runs whose p@30 scores are no less than 0.10, leaving us with 117 runs from the TREC 2012 Microblog track. For details about the implementation of the runs from the TREC 2012 Microblog track we refer to (Soboroff et al., 2012). The track continued in 2013, but with a different experimental setup, where participants used a shared API to retrieve documents that were subsequently re-ranked.

4.3. Baselines

We compare BurstFuseX to 3 data fusion baselines: 2 traditional unsupervised methods, i.e., CombSUM, CombMNZ, and a start-of-the-art supervised method, λ-Merge (Sheldon et al., 2011). As BurstFuseX utilizes burst information to boost the performance, we also compare BurstFuseX to 4 state-of-the-art burst-sensitive microblog search algorithms: time-based language model (TBLM) (Li & Croft, 2003), textual quality factor model with temporal query expansion (LM-T(qe)) (Massoudi et al., 2011), direct time-sensitive BM25 retrieval model (DIRECT-BM25 (mean)) (Dakka et al., 2012) and temporal tweet selection feedback method (TSF + QDRM) (Miyanishi et al., 2013b). All of these burst-sensitive algorithms first detect bursts

[7] We reorder all 2011 runs by retrieval score before fusing them.
[8] These 2011 and 2012 runs can be downloaded from http://trec.nist.gov.

```
< top>
< num> Number: MB023 < /num>
< title> Amtrak train service < /title>
< querytime> Tue Feb 08 20:04:25 +0000 2011 < /querytime>
< querytweettime> 35066441501900800 < /querytweettime>
< /top>
```

Fig. 3. Example topic, MB023, in the TREC 2011 Microblog track.

(or time-spans) based on the content (words) of the posts and then utilize the burst information to boost the retrieval performance. Our BurstFuseX detect bursts based on the fusion scores of posts rather than directly based on the content of the posts.

To illustrate the merits of detecting bursts from fusion scores, we implement an alternative algorithm, BurstFuseX$_{posts}$ (BurstFuseCombSUM$_{posts}$ and BurstFuseCombMNZ$_{posts}$), which detects bursts using the burst detection approach presented in (Lappas et al., 2009), using the content of posts, and then fuses the input rank lists using our fusion framework. In other words, the only difference between BurstFuseX and BurstFuseX$_{posts}$ is in the way they detecte bursts. To build the index of the dataset that some of our baselines require, we apply Porter stemming, tokenization, and stopword removal (using INQUERY lists) to posts using the Lemur toolkit.[9] The features and settings used for λ-Merge are briefly described in Appendix A.

4.4. Metrics and significance testing

For performance evaluation in our experiments we consider both minimally relevant and highly relevant posts relevant and use the official metric, p@30 (precision at rank 30). We also report on p@5, p@10, p@15 and MAP (mean average precision) scores. Given a query q, the precision at rank k metrics p@k ($k = 5, 10, 15, 30$ in our experiments) for this query can be simply calculated by:

$$\text{p@}k = \frac{\sum_{i=1}^{k} \text{rel}(d_i)}{k},$$

where rel(d) is a binary function that indicates whether the document d_i at rank i is relevant to the query q:

$$\text{rel}(d_i) = \begin{cases} 1 & \text{if } d_i \text{ is relevant to } q, \\ 0 & \text{otherwise.} \end{cases}$$

The p@k score for a test collection is the average p@k score of all test queries in the collection. Mean average precision (MAP) is a commonly used recall-oriented metric. For each relevant document in the result list we take the precision at the rank of the document. We sum over these precision values and divide it by the total number of relevant documents. This gives us the average precision (AP) for a query, which can be formulated as:

$$AP = \frac{\sum_{k=1}^{N} \text{p@}k \times \text{rel}(d_k)}{|\mathcal{R}|},$$

where $\mathcal{R}$ is the set of relevant documents for a given query, and N is the number of returned documents. Then, the MAP score for a test collection is obtained by taking the mean of AP scores over a set of test queries.

We use trec_eval[10] to compute the performance scores. We expect BurstFuseX to have a recall-enhancing effect. This may negatively impact very early precision, which is why we include p@5. But we hypothesize that we will see an increase in precision scores at lower ranks because of the expected boost in recall and the limited length of the lists being scored (only 30 items). For this reason we consider precision scores at multiple cut-offs (5, 10, 15, 30) as well as MAP. Statistical significance of observed differences between the performance of two runs is tested using a two-tailed paired t-test; we use ▲ (or ▼) to denote significant differences for $\alpha = .01$, or △ (and ▽) for $\alpha = .05$.

4.5. Training and optimization

Our BurstFuseX fusion method incorporates a single free parameter, μ in Eq. (5). The value of μ ($\in \{0, 0.1, \ldots, 1\}$) is set using 10-fold cross validation performed over the entire set of queries in the TREC 2011 Microblog track. In the learning phase, the performance of BurstFuseX is optimized with respect to MAP. In other words, the set of 49 queries is randomly partitioned into 10 equal size subsamples; the performance for a single test subsample (5 queries) is that attained using a value of μ that maximizes MAP performance over the remaining subsamples (44 queries). We repeat the experiment 10 times and report the average scores on the metrics. In each time, the subsamples are permuted until all the 49 queries were

[9] http://www.lemurproject.org.
[10] Like the runs, the evaluation script can be obtained from http://trec.nist.gov.

chosen once for the test set. The setting of BurstFuseX over the entire set of queries in the TREC 2012 Microblog track is the same as that in TREC 2011 Microblog track. Our baseline fusion methods, i.e., CombSUM, CombMNZ and λ-Merge, do not incorporate free parameters.

4.6. Experiments

We report on 8 main experiments in this paper. First, to understand the overall performance of BurstFuseX, we sample about 10% from the ranked lists produced by the participants in the TREC 2011 and 2012 Microblog tracks based on the lists' p@30 distribution, respectively: 18 out of the 174 runs in TREC 2011 and 18 out of the 117 runs in TREC 2012, 6 each with p@30 scores between 0.20 and 0.30 (Class 3), between 0.30 and 0.40 (Class 2), and over 0.40 (Class 1). We also randomly choose two runs from each class to construct Class 4. See Tables 3 and 4 for details of our sample runs from the TREC 2011 and 2012 Microblog track, respectively. Note that in our experiments, the runs in Class 1 in Tables 3 and 4 are actually the six best ones in the TREC 2011 and 2012 Microblog tracks, respectively. In every class, we use run1, run2, run3, run4, run5 and run6 to refer to the runs in descending order of p@30 score.

To understand the influence of bursts and see whether burst information is helpful to boost fusion performance and to which extent, we change the parameter μ in Eq. (5) from 0.0 to 1.0, which alters the degree to which burst-based and standard fusion information are to be mixed. Then, to understand the effect of the number of lists to be merged, we randomly choose k ($\in \{2, 4, 6, \ldots, 38\}$) lists from the 174 TREC 2011 Microblog lists and fuse them by BurstFuseX and the standard fusion methods. We repeat the experiments 20 times and report the average results as well as the corresponding standard deviation scores.

In order to understand the topic-level performance of BurstFuseX, we provide an analysis of topic-level performance against the standard fusion method it cooperates. Next, to determine how fast BurstFuseX can merge result lists, we again fuse k ($\in \{2, 4, \ldots, 30\}$) lists, and report and compare the average computing time required by BurstFuseX against that of the standard fusion methods. To understand whether BurstFuseX can improve over microblog search approaches that already incorporate time-sensitive search algorithms, we compare the performance of BurstFuseX, the standard fusion method it builds on and 5 time-sensitive baselines of searching microblogs.

To understand whether detecting bursts based on standard fusion scores works better than detecting based on the textual content of posts, we make a comparison between our fusion methods and those detecting bursts based on the textual contents of posts. We also compare burst-sensitive component lists to be fused and the fusion methods to see whether fusion can help to boost the retrieval performance. Finally, to understand whether BurstFuseX requires multiple result lists or if it can aid single runs that may not have taken time into account, we fuse only a single result list using BurstFuseX and compare the single result list and the output of BurstFuseX on that list.

As described in Section 3, in our experiments we use two unsupervised data fusion methods, CombSUM and CombMNZ, and one supervised method, λ-Merge, as representatives of the standard methods that can be integrated by BurstFuseX.

5. Results and analysis

In this section, we present our experimental results and perform an analysis. We follow the order of the research questions listed in Section 4.1. In particular, in Section 5.1 we examine the effectiveness of BurstFuseX on fusing the sample lists; in Section 5.2 we study the effect of using burst information and in Section 5.3 the effect of the number of lists on the overall performance; Section 5.4 reports on a topic level analysis; Section 5.5 is devoted to look at the runtime performance of

Table 3
Summary of sampled runs from the TREC 2011 Microblog track.

Class	Sampled runs	Performance
Class 1	Clarity1, waterlooa3, FASILKOM02, isiFDL, DFReeKLIM30, PRISrun1	$0.40 \leqslant$ p@30
Class 2	KAUSTRerank, ciirRun1, gut, dutirMixFb, normal, UDMicroIDF	$0.30 \leqslant$ p@30 < 0.40
Class 3	WESTfilext, LThresh, qRefLThresh, run3a, Nestor, uogTrLqea	$0.20 \leqslant$ p@30 < 0.30
Class 4	FASILKOM02, waterlooa3, gut, UDMicroIDF, run3a, qRefLThresh	$0.20 \leqslant$ p@30

Table 4
Summary of sampled runs from the TREC 2012 Microblog track.

Class	Sampled runs	Performance
Class 1	hitURLrun3, kobeMHC2, kobeMHC, uwatgclrman, hitQryFBrun4, kobeL2R	$0.40 \leqslant$ p@30
Class 2	QEWebFB, indri, KLIMLL, UNCRQE, gucasGenReg, KLIMLPLL	$0.30 \leqslant$ p@30 < 0.40
Class 3	FASILKOM01, IIEIR03, RUN2, expansion, IRSIISI, uwatgclrbase	$0.20 \leqslant$ p@30 < 0.30
Class 4	hitQryFBrun4, kobeL2R, KLIMLL, UNCRQE, IIEIR03, IRSIISI	$0.20 \leqslant$ p@30

BurstFuseX and in Section 5.6 we examine whether BurstFuseX is able to add anything in terms of performance on top of result lists produced by retrieval methods that already use temporal information; Section 5.7 provides a further analysis of the use of burst information in data fusion for microblog search; finally, Section 5.8 shows the performance of BurstFuseX on single result list.

5.1. Fusing the sample lists

We begin by addressing research questions i–iii. The performance of BurstFuseX and of the standard fusion methods X that it incorporates is detailed in Table 5, with scores based on the 10% sample runs from the TREC 2011 Microblog track, as mentioned in Section 4.6. It is clear from Table 5 that the performance of unsupervised data fusion and the corresponding burst-aware fusion methods, i.e., CombSUM, CombMNZ, BurstFuseCombSUM and BurstFuseCombMNZ, is better than that of the best result list that is used in the merging process (run1) for all classes and on almost all metrics. Many of these improvements are statistically significant. More importantly, in class 1 all of these methods beat the best recorded run (isiFDL) in the TREC 2011 Microblog track (e.g., the p@30 score for BurstFuseCombSUM is 0.5578 while that of the best run in the track is 0.4551), and even the standard fusion method it integrates, i.e., CombSUM, outperforms the best recorded run. Meanwhile, in class 1, the supervised data fusion methods λ-Merge and BurstFuseλ-Merge can also beat the best recorded run in terms of the official metric p@30. All of this demonstrates that data fusion strategies can help improve effectiveness in searching microblogs. One of the main reasons behind this is that various retrieval approaches often return very different irrelevant posts, but many of the same relevant posts.

It is worth noting that in most cases BurstFuseX outperforms the standard fusion method X that it incorporates for all classes and on nearly all metrics (MAP, p@10, p@15, p@30). Almost all of these improvements are substantial and statistically significant. For instance, when fusing the runs in class 4, the MAP and p@30 metrics of BurstFuseCombMNZ are 0.2883 and 0.4387, respectively, compared to only 0.2794 and 0.4048, respectively, for CombMNZ. This finding attests to the merits of incorporating burst information into data fusion and shows that using burst information can improve the performance of existing data fusion methods in terms of MAP and p@30.

Table 5
Retrieval performance on the 10% sample lists from the TREC 2011 Microblog track. Boldface marks the better performance between BurstFustX and the standard fusion method X that it incorporates; a statistically significant difference between the two is marked in the upper right hand corner as ▲ (or ▼) for $\alpha = .01$, or △ (and ▽) for $\alpha = .05$; a statistically significant difference with run1 is marked in the upper left hand corner using the same symbols; the best result per column is underlined.

	Class 1					Class 2				
	MAP	p@5	p@10	p@15	p@30	MAP	p@5	p@10	p@15	p@30
run1	.2210	.5918	.5673	.5347	.4551	.1457	.4612	.4143	.3714	.3571
run2	.2690	.5959	.5796	.5442	.4537	.1886	.4776	.4347	.3878	.3463
run3	.2318	.5755	.5367	.5034	.4401	.1525	.4041	.4143	.3878	.3408
run4	.2058	.5714	.5367	.4939	.4211	.1376	.3959	.3939	.3796	.3218
run5	.2575	.5673	.4980	.4721	.4211	.1688	.3878	.3633	.3605	.3136
run6	.2098	.5469	.5102	.4694	.4095	.1820	.4122	.3796	.3619	.3027
CombSUM	.3404	▲**.6245**	.5816	.5524	▲.4966	▲.2625	▲**.5306**	▲.4531	▲.4286	▲.3735
BurstFuseCombSUM	▲**.3563**▲	▲.6163	▲**.5959**△	▲**.5878**▲	▲**.5578**▲	▲**.2651**▲	△.4898▼	▲**.4694**▲	▲**.4553**▲	▲**.4344**▲
CombMNZ	▲.3385	▲.6245	.5755	.5524	▲.5020	▲.2581	▲**.5347**	▲.4592	▲.4354	▲.3789
BurstFuseCombMNZ	▲**.3528**▲	▲**.6286**	▲**.5959**▲	▲**.5918**▲	▲**.5517**▲	▲**.2587**▲	▲.5061▼	▲**.4735**▲	▲**.4567**△	▲**.4242**▲
λ-Merge	.2548	▼.5641	▼.5631	.5496	.4611	.1898	▽**.4641**	▲.4608	▲.4307	▲.3668
BurstFuseλ-Merge	▲**.2920**▲	▼**.5655**	**.5812**▲	▲**.5701**▲	▲**.5011**▲	▲**.2161**▲	▼.4384▼	▲**.4651**	▲**.4558**▲	▲**.4195**▲
	Class 3					Class 4				
run1	.1661	.4041	.3408	.2898	.2122	.2058	.5714	.5367	.4939	.4211
run2	.0997	.3429	.3000	.2653	.2095	.2098	.5469	.5102	.4694	.4095
run3	.1636	.3959	.3122	.2571	.2041	.1376	.3959	.3939	.3796	.3218
run4	.0753	.3265	.2735	.2585	.2034	.1820	.4122	.3796	.3619	.3027
run5	.0571	.2980	.2551	.2408	.2020	.1636	.3959	.3122	.2571	.2041
run6	.0994	.3510	.2735	.2408	.2016	.0753	.3265	.2735	.2585	.2034
CombSUM	▲.2150	▲**.4857**	▲**.4327**	▲.3837	▲.2952	.2795	▲**.6122**	▲.5327	▲.4721	▼.3918
BurstFuseCombSUM	▲**.2283**	▲.4408▼	▲.4184▽	▲**.3973**△	▲**.3388**▲	▲**.2863**▲	▲.5633▼	▲**.5449**△	▲**.5011**▲	▲**.4380**▲
CombMNZ	▲.2187	▲**.4898**	▲**.4327**	▲.3932	▲.2973	.2794	▲**.6000**	▲.5449	△.4830	.4048
BurstFuseCombMNZ	▲**.2313**△	▲.4531▼	▲**.4327**	▲**.4122**	▲**.3442**▲	▲**.2883**▲	▲.5796▼	▲**.5469**	△**.5043**▲	**.4387**▲
λ-Merge	▼.1450	▼**.3757**	▲.3709	▲.3431	▲.2801	▽.1950	▼**.4816**	▼.4708	▲.4628	.4038
BurstFuseλ-Merge	▼**.1513**▲	▼.3547▽	▲**.3810**△	▲**.3572**▲	▲**.3217**▲	▲**.2212**▲	▼.4609▼	▼**.4811**△	▲**.4937**▲	▲**.4303**▲

Interestingly, in Table 5 when we consider the p@5 scores, we see that BurstFuseX always outperforms the best single run but that it loses against the standard fusion method X on which it builds in most cases. One reason is that some relevant posts ranked very high in the lists being merged but not near any of the bursts are forced down the ranking. Another reason is that a very small number of irrelevant posts in the bursts are promoted to slightly higher ranks in the fused list. In contrast, the gain obtained from relevant posts that are ranked at the bottom of the runs but near bursts are clearly observed at bigger cut-offs, resulting in the improvements of p@10, p@15 and p@30 scores.

Additionally, from Table 5 we see that, in terms of MAP, BurstFuseCombSUM outperforms BurstFuseCombMNZ, and both of them outperform BurstFuseλ-Merge in Class 2 (0.2651, 0.2587, 0.2161, respectively). In other words, BurstFuseCombSUM outperforms BurstFuseCombMNZ, followed by BurstFuseλ-Merge. This is quite obvious in Class 1, Class 2 and Class 4 for instance. In terms of the standard fusion methods, CombSUM performs almost the same as CombMNZ without statistically significant differences. Both CombSUM and CombMNZ easily beat the supervised standard fusion method λ-Merge; in same cases, λ-Merge becomes worse than the best result list (run1) to be fused in the corresponding class. This may be due to over-fitting of λ-Merge.

As a sanity check, so as to confirm our observations about the performance of BurstFuseX, we also test BurstFuseX on the 10% sample runs from TREC 2012 Microblog track. We present the experimental results of BurstFuseX and the standard fusion method X it incorporates in Table 6. Clearly, our observations about the performance of BurstFuseX and the standard fusion methods when fusing the runs from TREC 2011 Microblog track are confirmed by the 2012 data. For instance, in Table 6 we see that all data fusion methods, both BurstFuseX and other ones, when fusing runs in Class 1, outperform the best result run of the TREC 2012 Microblog track, and almost all of these improvements are statistically significant in terms of the metrics listed in the table. Below, we only report experimental results for the TREC 2011 Microblog track test collection: the 2012 collection consistently yields the same overall results and trends.

5.2. The use of burst information

Next we address research question iv and examine the effect of using different amounts of burst information in our burst-aware fusion method. Put differently, we examine the impact of the free parameter μ in Eq. (5). Fig. 4 depicts the

Table 6
Retrieval performance on the 10% sample lists from the TREC 2012 Microblog track. Boldface marks the better performance between BurstFustX and the standard fusion method X that it incorporates; a statistically significant difference between the two is marked in the upper right hand corner as ▲ (or ▼) for $\alpha = .01$, or △ (and ▽) for $\alpha = .05$; a statistically significant difference with run1 is marked in the upper left hand corner using the same symbols; the best result per column is underlined.

	Class 1					Class 2				
	MAP	p@5	p@10	p@15	p@30	MAP	p@5	p@10	p@15	p@30
run1	.1685	.6102	.5729	.5254	.4695	.1078	.4508	.4475	.4305	.3655
run2	.1582	.5898	.5627	.5424	.4684	.1136	.4881	.4492	.4271	.3638
run3	.1526	.5729	.5407	.5322	.4610	.1010	.4712	.4322	.4023	.3599
run4	.1563	.6000	.5763	.5480	.4571	.1095	.4576	.4017	.3842	.3463
run5	.1566	.5831	.5390	.4949	.4435	.1079	.4136	.3966	.3706	.3390
run6	.1385	.5525	.5237	.5028	.4429	.0744	.3898	.3678	.3605	.3209
CombSUM	.1785	▲**.6271**	.5814	▲.5559	▲.5028	▲.1263	▲**.5322**	▲.4898	▲.4520	△.3797
BurstFuseCombSUM	▲**.2028**▲	▼.5797▼	**.5763**	▲**.5842**▲	▲**.5797**▲	▲**.1473**▲	▲.4983▼	▲**.4898**	▲**.4734**▲	▲**.4548**▲
CombMNZ	.1776	▲**.6339**	.5831	▲.5582	▲.5011	▲.1319	▲**.5492**	▲.5085	▲.4610	▲.3944
BurstFuseCombMNZ	▲**.2019**▲	.6000▼	**.5847**	▲**.5887**▲	▲**.5678**▲	▲**.1516**▲	▲.5186▼	▲**.4949**	▲**.4859**▲	▲**.4651**▲
λ-Merge	.1700	.6134	.5673	▲.5462	.4718	.1131	▲**.5124**	△.4583	.4367	.3698
BurstFuseλ-Merge	▲**.1843**△	▼**.5734**▼	**.5730**	▲**.5675**▲	▲**.5184**▲	▲**.1326**▲	▲.4878▼	▲**.4766**▲	▲**.4638**▲	▲**.4281**▲
	Class 3					Class 4				
run1	.0694	.3322	.3102	.2983	.2712	.1566	.5831	.5390	.4949	.4435
run2	.0661	.3356	.3390	.3062	.2678	.1385	.5525	.5237	.5028	.4429
run3	.0724	.3390	.3390	.3073	.2542	.1010	.4712	.4322	.4023	.3599
run4	.0658	.3017	.2864	.2938	.2480	.1095	.4576	.4017	.3842	.3463
run5	.0626	.2949	.2661	.2520	.2282	.0661	.3356	.3390	.3062	.2678
run6	.0343	.1831	.2068	.2147	.2367	.0343	.1831	.2068	.2147	.2367
CombSUM	▲.1082	▲**.5051**	▲**.4576**	▲.4158	▲.3249	▼.1360	▼**.5559**	▽.5288	.4949	▼.3955
BurstFuseCombSUM	▲**.1228**▲	▲.4508▼	▲.4424▽	▲**.4260**△	▲**.3904**▲	**.1584**▲	▼.5220▼	▼**.5102**▽	**.4938**	▲**.4774**▲
CombMNZ	▲.1163	▲**.5186**	▲.4746	▲.4362	▲.3508	▽.1456	▼**.5593**	.5356	.5062	▽.4260
BurstFuseCombMNZ	▲**.1326**▲	▲.4780▼	▲.4797	▲**.4554**△	▲**.4147**▲	△**.1661**▲	▼.5254▼	▼**.5300**	▲**.5153**	▲**.4842**▲
λ-Merge	▲.1075	▲**.4983**	▲.4458	▲.4037	▲.3352	▼.1301	▼**.5416**	▼.5119	.4863	▼.3847
BurstFuseλ-Merge	▲**.1104**▼	▲.4684▼	▲**.4482**	▲**.4267**▲	▲**.3982**▲	**.1488**▲	▼.5362	▼**.5107**	**.4954**	**.4477**▲

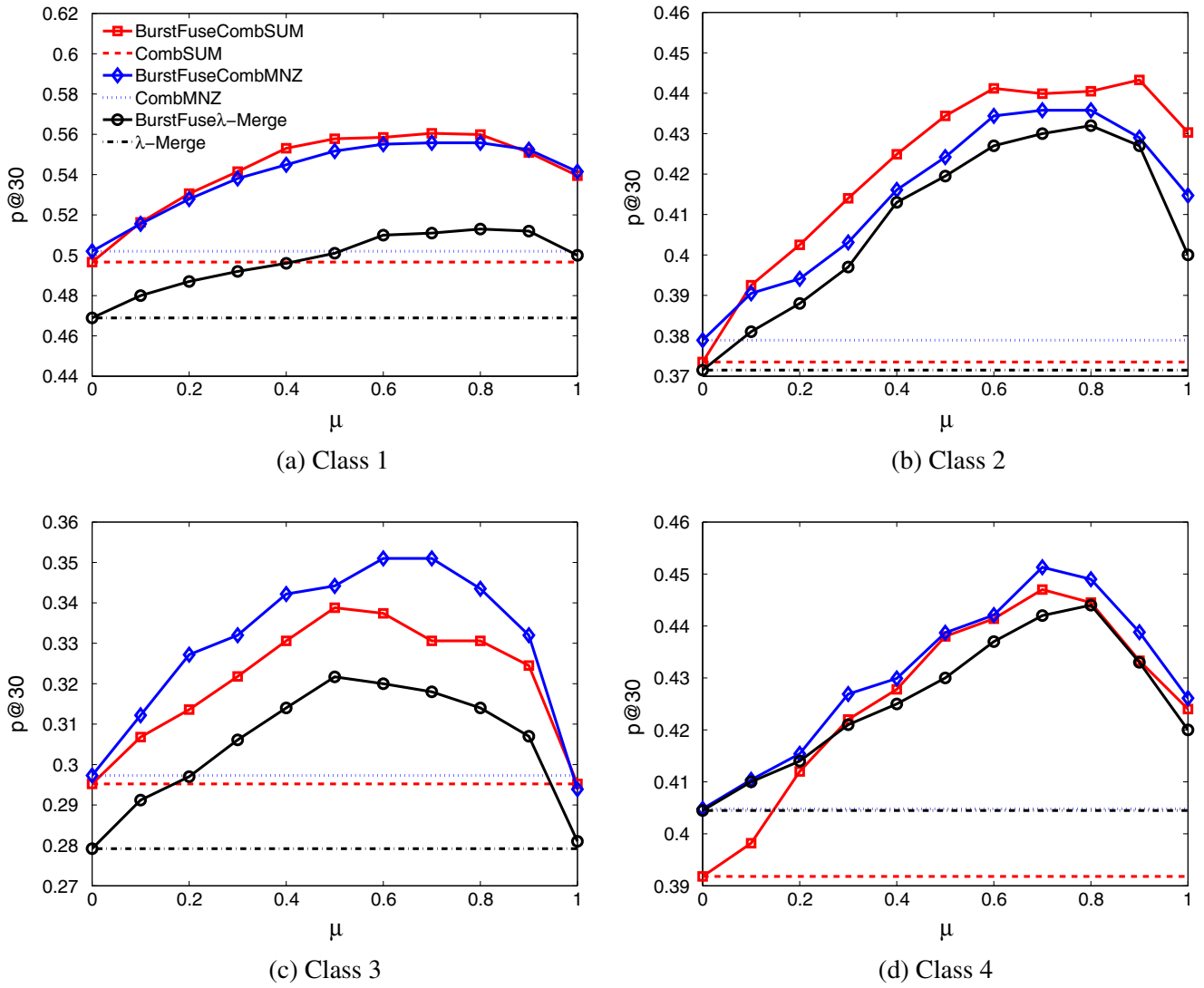

Fig. 4. Effect of varying the value of μ on the P@30 performance of BurstFuseX when merging lists in (a) Class 1, (b) Class 2, (c) Class 3 and (d) Class 4. When $\mu = 0$, BurstFuseX amounts to the standard fusion method X that it integrates. More weight is put on burst information with higher value of μ. Note: figures are not to the same scale.

p@30 performance curves for BurstFuseX and the corresponding standard fusion methods it integrates when fusing result lists in Class 1, Class 2, Class 3 and Class 4, respectively. For $\mu = 0$, BurstFuseX amounts to the standard fusion method X that it integrates; more weight is put on burst information with higher values of μ; for $0 < \mu < 1$, the standard fusion scores of posts (according to method X) as well as the burst information are utilized for fusing the lists.

It is worth noting in Fig. 4 that when fusing lists in different quality classes, the official metric p@30 scores of BurstFuseX ($\mu > 0$) are usually higher than those of the standard fusion method X it incorporates ($\mu = 0$), especially when $\mu = 0.6,\ 0.7$. For instance, in Class 4 the p@30 performance of BurstFuseCombSUM peaks at $\mu = 0.7$ with the score of 0.4451, while that of the standard fusion method it integrates, CombSUM achieves only 0.3918. As we observed before, BurstFuseX works better when it integrates one of the unsupervised standard fusion methods than the supervised fusion method λ-Merge.

In addition, it is clear from Fig. 4 that when the quality of the result lists as a whole is higher, it will be more useful to utilize burst information. Fig. 4 shows that even if $\mu = 1.0$, BurstFuseX still outperforms the standard fusion method it incorporates in high quality classes, like Class 1 and Class 2. But when the quality of the result lists as a whole becomes lower, the positive impact of BurstFuseX is reduced. We provide a further analysis of the use of burst information in data fusion and single retrieval microblog search algorithms in Sections 5.7 and 5.8.

5.3. Effect of the number of lists to be merged

We have already seen that BurstFuseX outperforms the standard fusion methods it incorporates when fusing 6 lists in different quality classes. We now address research question v and explore the effect on the performance of BurstFuseX of varying the number of lists being merged, in terms of MAP, p@5, p@15 and p@30. In Fig. 5, we randomly choose

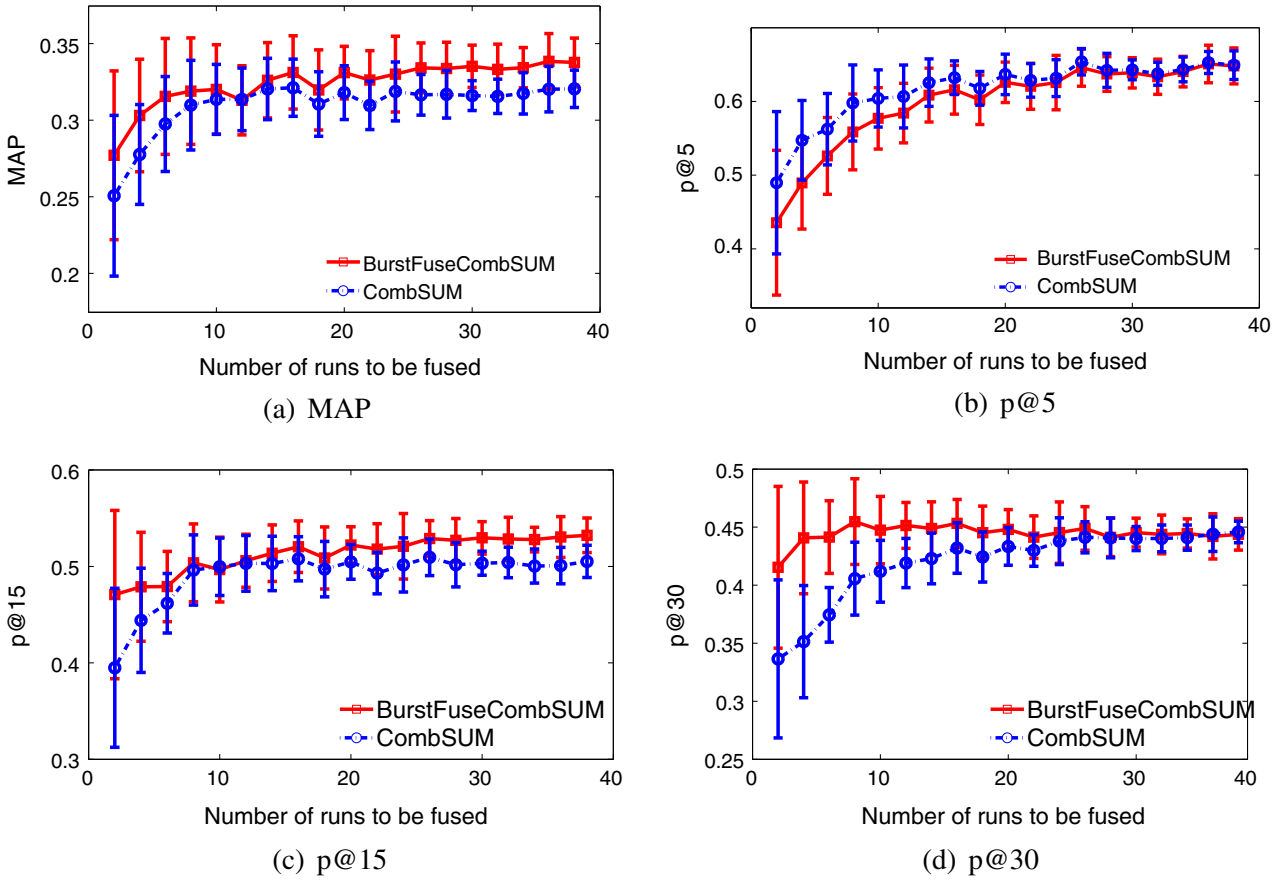

Fig. 5. Effect of the number of lists to be merged, k, on (a) MAP (b) p@5 (c) p@15 and (d) p@30. The scores of performance are with the corresponding standard deviation. Note: figures are not to the same scale.

$k \in \{2, 4, 6, \ldots, 38\}$ lists from the 174 lists made available by the TREC 2011 Microblog track and then fuse them. For each k, we repeat the experiment 20 times and report on the average scores as well as the standard deviation. We use CombSUM and BurstFuseCombSUM as a representative example; for the other combinations with a standard fusion method X qualitatively similar results can be observed.

As can be seen in Fig. 5, with fewer than 12 lists to be merged by either BurstFuseX or the standard fusion method it builds on, the addition of one more list tends to lead to performance increases across all metrics. Beyond 12 lists, the performance gains of additional lists tend to level off. This is because despite the fact that with more lists, on average we may see more high quality lists, more low quality lists may show up as well. Unlike the MAP, p@15 and p@30 performance of BurstFuseCombSUM where it always enhances that of CombSUM, the early precision p@5 performance of BurstCombSUM is worse than that of CombSUM especially when fewer lists are merged. This observation is consistent with those in both Tables 5 and 6. Performance gains in terms of p@15 of BurstFuseCombSUM and CombSUM continue even when more than 16 lists are being fused. In contrast, in terms of p@30, when the number of lists to be merged increases, the gain of BurstFuseCombSUM over CombSUM decreases. As more lists are being fused, more noise is being brought in, especially with posts ranked lower.

5.4. Topic-level analysis

Next, we take a closer look at per query improvements of BurstFuseX over the underlying standard fusion method X, thereby addressing research question vi. For brevity, we only consider BurstFuseCombSUM as a representative to report all the queries (topics) performance differences against that of the standard fusion method it incorporates, and we only consider runs in Class 1, Class 2, Class 3 and Class 4, respectively. The results for other instances of BurstFuseX are qualitatively similar.

Fig. 6 shows the per query performance differences in terms of AP, p@5, p@15 and p@30, respectively, between BurstFuseCombSUM and CombSUM. Overall, gains by BurstFuseCombSUM over CombSUM outnumber losses for p@15 and p@30 as well as MAP, but not for very early precision, i.e., p@5. Gains by BurstFuseCombSUM over CombSUM are due mainly to topics that are discussed only in very specific time intervals. Examples include topics MB010 (Egyptian protesters attack museum), MB011 (Kubica crash) and MB015 (William and Kate fax save-the-date). Invariably, for such topics we found evidence of the intuition depicted in Fig. 1: posts that are ranked low in a small number lists but that are pushed into the final merged list by

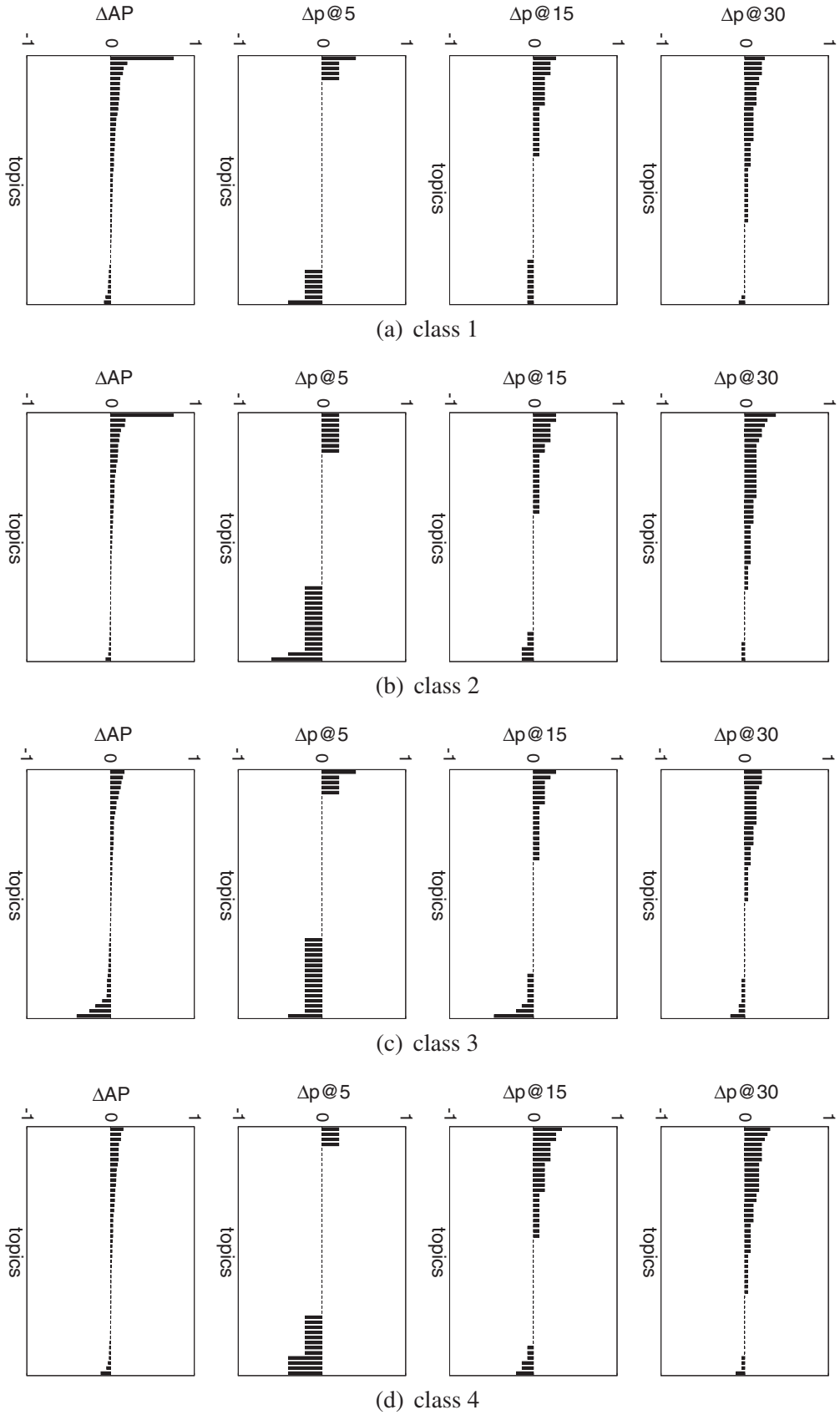

Fig. 6. Per topic performance differences of BurstFuseCombSUM against CombSUM. The figures shown are for the 2011 Microblog track sampled runs in Class 1, Class 2, Class 3 and Class 4, for AP, p@5, p@15 and p@30 difference. A bar extending to the right of the center of a plot indicates that BurstFuseCombSUM outperforms CombSUM, and vice versa for bars extending to the left of the center.

BurstFuseX because they are central to a burst. For instance, in response to topic MB010 (Egyptian protesters attack museum), post #30354903104749568 is ranked near the bottom in only two lists (at ranks 26 and 27 in runs clarity1 and DFReekLIM30, respectively). Because many posts for the topic were generated around the same time interval (January 26–29, 2011, when the event happened) and are ranked highly in many lists to be fused, post #30354903104749568 is rewarded and ranked as high as top 6 in the merged list.

Topics for which BurstFuseCombSUM cannot beat ComSUM tend to be quite general and unrelated to any specific time windows. Examples include topics MB023 (Amtrak train service) and MB027 (reduce energy consumption). For a very small number of queries, BurstFuseCombSUM's performance, in terms of MAP or p@30 is worse than that of CombSUM. One reason that we observed for this phenomenon is that a very small number of posts are not relevant to the topics even if they are central to the bursts according to their timestamps, and hence they should not be rewarded. An example here is topic MB031 (Special Olympics athletes). In response to this topic, result lists to be fused ranked some irrelevant posts highly, but these posts are still in the bursts, which results in promoting these posts to high ranks, which hurts the performance.

5.5. Run-time analysis

We now turn to research question vii and examine the run-times of BurstFuseX. In particular, we explore what the added costs in terms of run-time of BurstFuseX is on top of the standard fusion methods that it incorporates. Our implementation of BurstFuseX is developed in C++ and the experiments are run on a 10.6.8 MacBook Pro computer with 4 GB memory and a 2.3 GHz Intel core i5 processor. In Table 7, we randomly choose $k \in \{2, 4, 6, 8, 12, 18, 24, 30\}$ lists from the 174 lists available from the TREC 2011 Microblog track. For each k, we repeat the following experiment 20 times: sample a query from the TREC 2011 Microblog track test set, run fuse k result lists for the query (using CombSUM, CombMNZ, λ-Merge as well as BurstFuseCombSUM, BurstFuseCombMNZ and BurstFuseλ-Merge), record the wall clock time. The results are recorded in Table 7 and plotted in Fig. 7.

As can be seen in Table 7 and Fig. 7, the overhead of running BurstFuseX over simply running the standard fusion method X is very limited, but increases with the number of lists to be merged, especially when BurstFuseX incorporates with CombSUM and CombMNZ. BurstFuseCombSUM and BurstFuseCombMNZ merge the lists within 0.01 s when given 30 result lists and within 0.001 s when fusing two lists. In contrast, however, compared to any of the fusion methods, BurstFuseλ-Merge has to spend more time, which is almost the same as that of the fusion method it builds on. In addition, it is worth noting in Fig. 7 that in many cases, as the number of lists to be fused increases, the time spent on fusing is almost linear for BurstFuseX and the standard fusion method as well. For instance, the time needed to fuse 8 lists by BurstFuseCombSUM is nearly double the time needed for fusing 4 lists (2.96e−3 s and 1.50e−3 s, respectively).

5.6. Effect of fusing time-sensitive result lists

BurstFuseX uses temporal information in an essential way. Research question viii asks what happens when BurstFuseX fuses result lists that have been generated using temporal information themselves? That is, is there anything left to gain by using BurstFuseX? To answer this question, we explore the performance of BurstFuseX using five result lists that themselves consider temporal information: isiFDRML (Metzler & Cai, 2011; Metzler et al., 2012), DFReeKLIM30 (Horn et al., 2011), Wise2ndRun (Wei et al., 2011), ICTNET11MBR3 (Cao et al., 2011) and UDMicroIDFD (Amati et al., 2011). We use BurstFuseX as well as the standard fusion methods, CombSUM, CombMNZ and λ-Merge to fuse those result lists, and report the comparison results.

Table 7
Time spent on fusing lists by different aggregation methods. Recorded in seconds with standard deviations (std).

	Number of lists							
	2	4	6	8	12	18	24	30
CombSUM	3.06e−4	5.01e−4	5.76e−4	9.33e−4	1.03e−3	1.98e−3	2.77e−3	3.37e−3
std	1.13e−5	1.27e−5	2.57e−5	3.61e−5	6.93e−5	6.49e−5	7.02e−5	7.50e−5
CombMNZ	3.06e−4	5.01e−4	5.76e−4	9.33e−4	1.03e−3	1.99e−3	2.79e−3	3.38e−3
std	1.13e−5	1.27e−5	2.57e−5	3.61e−5	6.93e−5	6.52e−5	6.98e−5	7.01e−5
λ-Merge	1.15	2.55	3.82	5.24	7.78	12.03	16.32	20.74
std	1.22e−1	2.3e−1	5.82e−1	6.74e−1	8.03e−1	1.09	1.13	1.18
BurstFuseCombSUM	9.12e−4	1.50e−3	1.84e−3	2.96e−3	3.43e−3	6.69e−3	9.52e−3	1.38e−2
std	3.42e−5	3.94e−5	7.91e−5	7.41e−5	8.72e−5	1.88e−4	2.66e−4	2.84e−4
BurstFuseCombMNZ	9.12e−4	1.50e−3	1.84e−3	2.96e−3	3.43e−3	6.69e−3	9.52e−3	1.38e−2
std	3.42e−5	3.94e−5	7.91e−5	7.41e−5	8.72e−5	1.88e−4	2.66e−4	2.84e−4
BurstFuseλ-Merge	1.16	2.56	3.85	5.25	7.79	12.04	16.34	20.77
std	1.37e−1	2.57e−1	5.11e−1	6.30e−1	7.78e−1	1.10	1.02	2.41

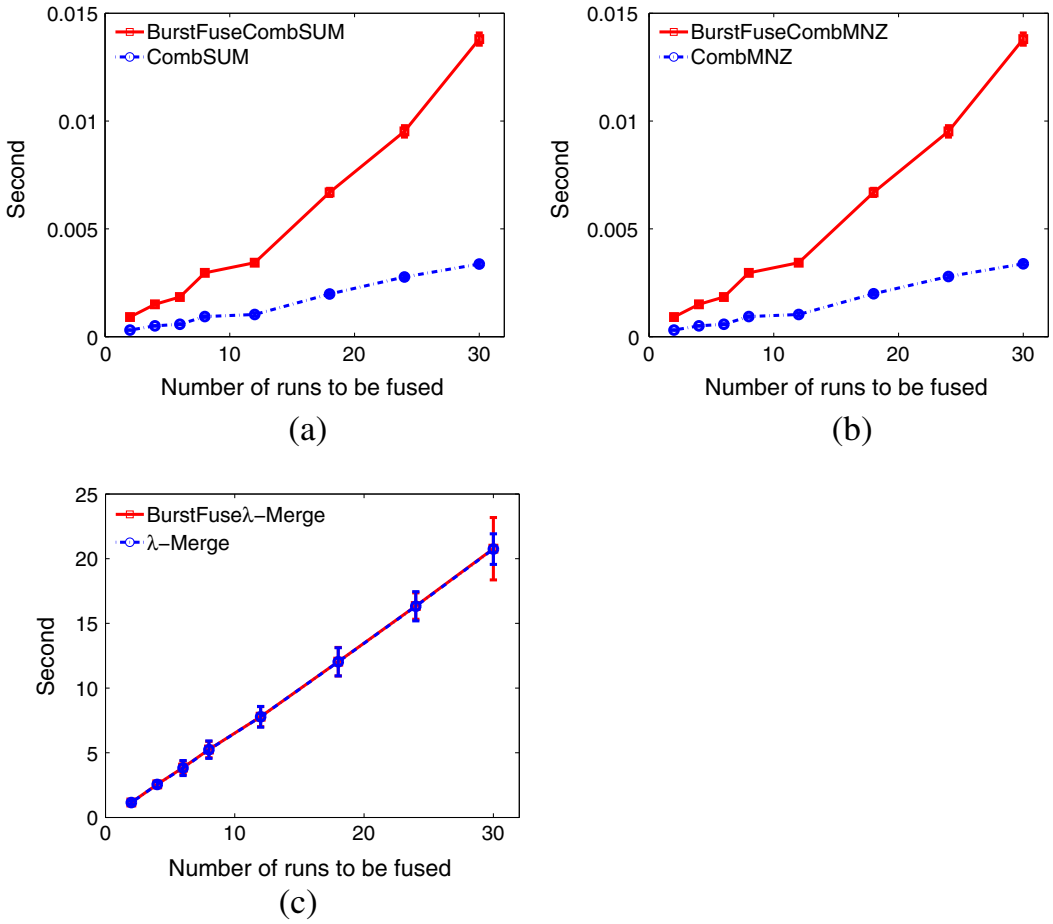

Fig. 7. Run-times of BurstFuseX against the standard fusion method it builds on with standard deviation. From left to right: (a) BurstFuseCombSUM against CombSUM, (b) BurstFuseCombMNZ against CombMNZ, and (c) BurstFuseλ-Merge against λ-Merge. Note: figures are not to the same scale.

Table 8 shows the result of the comparisons between BurstFuseX and the five time-sensitive result lists. Obviously, CombSUM and CombMNZ perform on a par with the best result list (isiFDRML). For all metrics but p@5, BurstFuseCombSUM and BurstFuseCombMNZ outperform the best result run as well as the standard fusion method they incorporate; many of the improvements are statistically significant. This illustrates that exploring time information in data fusion has a different effect than utilizing time information in an individual ranking function, an effect that can lead to performance increases. One of the main reasons behind this is that posts within intervals in which many relevant posts appear can only be confirmed to be

Table 8
Retrieval performance on 5 time-sensitive result lists. Boldface marks the better performance between BurstFustX and the standard fusion method X that it incorporates; a statistically significant difference between the two is marked in the upper right hand corner as ▲ (or ▼) for $\alpha = .01$, or △ (and ▽) for $\alpha = .05$; a statistically significant difference with the best single run (isiFDRML) is marked in the upper left hand corner using the same symbols; the best result per column is underlined.

	MAP	p@5	p@10	p@15	p@30
isiFDRML	.2326	.6286	.5633	.5374	.4442
DFReeKLIM30	.2318	.5755	.5367	.5034	.4401
Wise2ndRun	.1971	.4980	.4612	.4231	.3639
ICTNET11MBR3	.1863	.4490	.3735	.3619	.3054
UDMicroIDFD	.1428	.3224	.3143	.3075	.2687
CombSUM	.2397	**.6280**	.5673	.5401	.4469
BurstFuseCombSUM	▲**.3053**▲	▼.5837▼	▲**.5714**	▲**.5701**▲	▲**.4966**▲
CombMNZ	△.2421	**.6284**	.5780	▼.5048	△.4578
BurstFuseCombMNZ	▲**.3204**▲	.6245	▲**.5878**	▲**.5755**▲	▲**.5116**▲
λ-Merge	▼.2148	▼**.5539**	▼.5144	▼.4873	▼.3894
BurstFuseλ-Merge	▼**.2271**▲	▼.5501	▼**.5257**△	▼**.5187**▲	▽**.4335**▲

relevant by gathering data from multiple lists, time-sensitive or not. Finally, neither λ-Merge nor BurstFuseλ-Merge can beat the best time-sensitive result list; as before, BurstFuseλ-Merge does improve over λ-Merge.

5.7. Further analysis of using burst information

We continue with research question viii and provide a further analysis of the use of burst information in the setting of microblog search. More specifically, the component lists that we consider next are generated using a temporal model for microblogs (TMM) (Choi & Croft, 2012), a textual quality factor model with temporal query expansion (LM-T(qe)) (Massoudi et al., 2011), a direct time-sensitive BM25 retrieval model (DIRECT-BM25 (mean)) (Dakka et al., 2012) and a temporal tweet selection feedback method (TSF + QDRM) (Miyanishi et al., 2013b).

Table 9 shows a comparison between fusion methods and the component lists that are burst-sensitive. As can be seen in the table, except for the performance of λ-Merge, which is slightly worse than that of the best component list, all other fusion methods can boost retrieval performance, especially for the fusion methods that we propose in this paper. This finding again underlines the merit of fusion for searching microblog posts and of using bursts information in the fusion step.

In Table 9 we also compare our BurstFuseX with CombSUM, CombMNZ, λ-Merge, and BurstFuseX$_{posts}$, where BurstFuseX$_{posts}$ is the fusion method that detects bursts based on the content of posts by the detection approached in (Lappas et al., 2009) and then integrates burst information into the fusion process. Clearly, BurstFuseX can still significantly enhance the retrieval performance in all cases and outperforms all standard fusion methods and the best component run, in contrast with BurstFuseX$_{posts}$. We also see that detecting bursts based on one of the standard fusion methods, CombSUM or CombMNZ, works better than detecting bursts based on the content of posts.

Table 9
Performance on 4 burst-sensitive result lists. Boldface marks the best result per metric; the best score of component lists per metric is underline; a statistically significant difference between a fusion method and the best component list is marked in the upper left hand corner of the fusion score; a statistically significant difference between BurstFuseX$_{posts}$ and BurstFuseX is marked in the upper right hand corner of the BurstFuseX score.

	MAP	p@5	p@10	p@15	p@30
TSF + QDRM	.2834	.6220	.6856	.6279	.5368
DIRECT-BM25 (mean)	.2798	.6187	.6725	.6320	.5133
LM-T (qe)	.2346	.5836	.5648	.5178	.4471
TBLM	.2231	.5742	.5433	.5017	.4395
CombSUM	△.2962	△.6395	▲.6973	▲.6482	▲.5513
BurstFuseCombSUM$_{posts}$	△.3027	△.6398	▲.6979	▲.6487	▲.5557
BurstFuseCombSUM	△**.3047**	△**.6408**	▲**.6983**	▲**.6497**	▲**.5613**△
CombMNZ	△.2948	△.6350	△.6918	△.6347	△.5420
BurstFuseCombMNZ$_{posts}$	△.3015	△.6373	△.6950	△.6447	△.5433
BurstFuseCombMNZ	△.3027	△.6398	▲.6972	▲.6443	▲.5524▲
λ-Merge	.2806	.6212	▽.6768	▼.6178	▼.5243
BurstFuseλ-Merge$_{posts}$	.2854	.6275	.6825	.6233	.5314
BurstFuseλ-Merge	.2942	.6387	.6894	.6326	▽.5428

Table 10
Performance of BurstFuseX on individual result lists in Class 1. None of the differences between BurstFuseX and the corresponding single input result list is statistically significant.

	MAP	p@5	p@10	p@15	p@30
run1	.2590	.5959	.5796	.5442	.4537
BurstFuse applied to run1	.2593	.5959	.5837	.5442	.4537
run2	.2575	.5673	.4980	.4721	.4211
BurstFuse applied to run2	.2577	.5673	.5000	.4721	.4211
run3	.2318	.5755	.5367	.5034	.4401
BurstFuse applied to run3	.2319	.5755	.5367	.5048	.4401
run4	.2210	.5918	.5673	.5347	.4551
BurstFuse applied to run4	.2210	.5918	.5673	.5347	.4551
run5	.2098	.5469	.5102	.4694	.4095
BurstFuse applied to run5	.2098	.5469	.5102	.4707	.4095
run6	.2058	.5714	.5367	.4939	.4211
BurstFuse applied to run6	.2062	.5755	.5367	.4952	.4211

5.8. Performance of BurstFuseX on single result list

Finally, to address our final research question, ix, and understand whether BurstFuseX requires multiple result lists or whether it can aid single runs that may not have taken time into consideration, we feed BurstFuseX single result lists and compare the output against the single input list.

Table 10 shows the results on the result lists in class 1; results on other result lists are qualitatively similar. As can be seen in the table, the retrieval performance of BurstFuseX is almost the same as that of the input result list and the difference between them is not statistically significant for any of the metrics. The main reason why BurstFuseX cannot significantly beat the input result list is that detecting bursts within a small set of documents (i.e., those contained in a single result list) is challenging.

6. Conclusion

Various features of microblog posts make searching such posts a real challenge: their limited length, their dynamic nature, the creative language usage and their highly contextualized nature. However, the special nature of microblog posts also offers unique opportunities. In this paper, we have focused on utilizing one such special feature for boosting the performance of search algorithms for microblog posts. We have proposed a data fusion approach, BurstFuseX, that fuses result lists based in part on the bursty nature of many discussions on microblog platforms. Our approach is based on integrating information generated by a standard fusion method, such as CombSUM, CombMNZ or λ-Merge, detecting bursts of posts across the lists being fused, and rewarding posts that are published in or near a burst containing highly ranked posts. Our experimental results show that detecting bursts and then using burst information into a standard fusion method can enhance the retrieval performance compared to the standard fusion method it integrates, in terms of mean average precision as well as precision-oriented measures. Our new fusion method has a strong recall-enhancing effect; compared to the standard fusion method it incorporates, this comes at a small price in terms of a small drop in very early precision measures such as p@5. Our experimental results also show that our BurstFuseX method can significantly outperform burst or time-sensitive retrieval models and models that detect bursts based on the content of posts.

As to future work, we have only explored data fusion techniques in microblog search. But data fusion can be, and has been, applied in a variety of areas in IR, like federated search (Crestani & Markov, 2013; Shokouhi & Si, 2011), cross-lingual search (Si et al., 2008), and finding groups of knowledgeable experts (Liang & de Rijke, 2013). How to apply our burst-aware data fusion in these other areas is an open research question. One other avenue for future work is to integrate temporal information into web search—for so-called fresh results (Lefortier, Serdyukov, & de Rijke, 2014).

Acknowledgements

We would like to thank our reviewers for valuable comments and suggestions that helped to improve the paper.

This research was partially supported by the China Scholarship Council, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements nr 288024 (LiMoSINe) and nr 312827 (VOX-Pol), the Netherlands Organisation for Scientific Research (NWO) under project nrs 727.011.005, 612.001.116, HOR-11-10, 640.006.013, the Center for Creation, Content and Technology (CCCT), the QuaMerdes project funded by the CLARIN-nl program, the TROVe project funded by the CLARIAH program, the Dutch national program COMMIT, the ESF Research Network Program ELIAS, the Elite Network Shifts project funded by the Royal Dutch Academy of Sciences (KNAW), the Netherlands eScience Center under project number 027.012.105 the Yahoo! Faculty Research and Engagement Program, the Microsoft Research PhD program, and the HPC Fund.

Appendix A. Detailing λ-Merge

In this appendix we detail one of the standard fusion methods we use, λ-Merge. To be able to define the fusion score for document d in response to query q according to λ-Merge, we need to consider the sum of weighting the individual document scores in each list by the weight of the corresponding list:

$$g(d;q) = \sum_{m} \alpha_m \cdot f(x_d^{L_m};\boldsymbol{\theta}), \tag{A.1}$$

where α_m is the weight of list L_m, $f(x_d^{L_m};\boldsymbol{\theta})$ is the scoring function for d in L_m, with parameters $\boldsymbol{\theta}$. We adapt a linear function for $f(x_d^{L_m};\boldsymbol{\theta})$, due to its widespread use (Atrey, Hossain, Saddik, & Kankanhalli, 2010), such that:

$$f(x_d^{L_m};\boldsymbol{\theta}) = \sum_{n} \theta_n \cdot x_{n_d}^{L_m},$$

where θ_n, the n-th dimension of vector $\boldsymbol{\theta}$, is the weight of the n-th feature, and $x_{n_d}^{L_m}$, the n-th dimension of $x_d^{L_m}$, is the value of the n-th feature of d in L_m.

Table A.11
Features used for λ-Merge.

Feature	Gloss
Query-post level	
Rank	Inverse of the rank of a post over the number of returned documents
Rankers	Percentage of rankers a post appears in
IsTop-*N*	If a post is within the top-*N* results
Post level	
Link	If a post has links
Hashtag	If a post has hashtags
Retweet	If a post has retweets
Density	A post's content quality
Capitalization	A post's textual quality
Length	A post's length deviation from the median length
Post-burstiness	A post's score associated with bursts (The definition of bursts can be found in Section 3.2)

Now, writing C to denote the smoothed objective, according to λ-Merge the parameters α_m and θ_n can be updated based on the gradients $\partial C/\partial\alpha_m = \sum_d(\partial C/\partial g(d;q)) \cdot (\partial g(d;q)/\partial\alpha_m)$ and $\partial C/\partial\theta_n = \sum_d(\partial C/\partial g(d;q)) \cdot (\partial g(d;q)/\partial\theta_n)$, respectively. Then, $\partial C/\partial g(d;q)$ is defined as:

$$\partial C/\partial g(d;q) = \sum_e |\Delta_{de}|\{\mathbb{I}_{d\succ e} - 1/(1+\exp(g(e;q)-g(d;q)))\},$$

where $|\Delta_{de}|$ is the absolute change in the performance metric if document d and e were swapped in the current ranking, and the indicator $\mathbb{I}_{d\succ e}$ is 1 when d is judged more relevant than e, 0 otherwise. As explained in the main text, MAP is the metric on which we focus; hence, we optimize λ-Merge for MAP.

Let r_d and r_e denote the rank positions of d and e in L_m. Assume that d and e are misranked by current function g, i.e., $r_d > r_e$ but the relevance level of $d, l(d)$, is larger than that of $e, l(e)$, then

$$|\Delta_{de}| = \frac{1}{R}\left(\sum_{k=r_e}^{r_d} l(k)P@k - \sum_{k=r_e}^{r_d} l'(k)P'@k\right).$$

Here, R is the number of relevant documents for that query, $l(k)$ is the relevance level of the document at rank position k, $P@k$ is the precision at rank k, and $l'(k)$ is the relevance value after the documents at positions r_d and r_e being swapped. Mathematically, the remaining derivatives can be presented as: $\partial g(d;q)/\partial\alpha_m = f(x_d^{L_m};\boldsymbol{\theta})$, and $\partial g(d;q)/\partial\theta_n = \sum_m \alpha_m \cdot x_{n_d}^{L_m}$. After training, the parameters $\boldsymbol{\theta}$ and the weight α_m of each list are obtained. Then we employ max–min normalization to $g(d;q)$, such that:

$$F_{\lambda-Merge}(d;q) := \frac{g(d;q) - \min(g_q)}{\max(g_q) - \min(g_q)}, \tag{A.2}$$

where $\min(g_q)$ and $\max(g_q)$ are the minimum and maximum value of g in response to q. For further details about λ-Merge we refer to (Donmez, Svore, & Burges, 2008; Sheldon et al., 2011).

Before training λ-Merge, a number of features have to be extracted for $d \in \mathcal{C}_{\mathcal{L}}$. Table A.11 lists the features used to construct our version of λ-Merge. We identify ten features, extracted from two levels: query-post level and post level; all features are represented by either binary or real numbers. At the query-post level, following Tsai et al. (2008), we use three features: *Rank*, *Rankers* and *IsTop-N*. *Rank* is defined in Eq. (1). *Rankers* is the number of ranked lists in which the post appears divided by the total number of lists to be merged. *IsTop-N* is a binary feature to indicate if this document is within the top-*N* results in the list. At the post level, we extract features capable of indicating the quality of a post (Macdonald et al., 2011; Ounis et al., 2011); the post features include *Link*, *Hashtag*, *Retweet* to indicate if the document contains links, hashtags, and retweets. The post-level features also consist of content quality indicators of a post (*Density*, *Capitalization* and *Length*) (Lee et al., 2001; Weerkamp & de Rijke, 2012). We also extract a feature that we call "post-burstiness" based on bursts. *Density* of a post is defined as the sum of $tf \cdot idf$ values of non-stopwords, divided by the number of stopwords they are apart, squared (Lee et al., 2001). As in Eq. (A.2) we use max–min normalization to normalize the scores. The *Capitalization* score of d is obtained by determining the fraction of sentences in d that have a leading capital, seeing to which degree this deviates from the median and then applying max–min normalization (Weerkamp & de Rijke, 2012). The *Length* score of d is obtained by considering the deviation from the median length of posts in the collection. The *Post-burstiness* score of d is obtained by utilizing burst information.

References

Ahmad, N., & Beg, M. M. S. (2002). Fuzzy logic based rank aggregation methods for the world wide web. In *Proceedings of the international conference on artificial intelligence in engineering and technology* (pp. 363–368).

Amati, G., Amodeo, G., Bianchi, M., Celi, A., Nicola, C. D., Flammini, M., et al. (2011). FUB, IASI-CNR, UNIVAQ at TREC 2011 microblog track. In *Proceedings of the text retrieval conference*.

Aslam, J. A., & Montague, M. (2001). Models for metasearch. In *SIGIR'01* (pp. 276–284).
Atrey, P., Hossain, M., Saddik, A. E., & Kankanhalli, M. (2010). Multimodal fusion for multimedia analysis: a survey. *Multimedia Systems, 16*(6), 345–379.
Bandyopadhyay, A., Mitra, M., & Majumder, P. (2011). Query expansion for microblog retrieval. In *TREC'11*.
Beg, M. M. S. (2004). Parallel rank aggregation for the world wide web. In *Proceedings of the intelligent sensing and information processing* (pp. 385–390).
Beitzel, S. M., Jensen, E. C., Chowdhury, A., Frieder, O., Grossman, D. A., & Goharian, N. (2003). Disproving the fusion hypothesis: An analysis of data fusion via effective information retrieval strategies. In *SAC'03* (pp. 823–827).
Bruno, E., & Marchand-Maillet, S. (2009). Multiview clustering: A late fusion approach using latent models. In *SIGIR'09* (pp. 736–737).
Cao, P., Gao, J., Yu, Y., Liu, Y., & Cheng, X. (2011). ICTNET at microblog track TREC 2011. In *Proceedings of the text retrieval conference*.
Chang, Y., Dong, A., Kolari, P., Zhang, R., Inagaki, Y., Diaz, F., et al (2013). Improving recency ranking using twitter data. *ACM Transactions on Intelligent Systems and Technology, 4*(1), 4:1–4:24.
Chen, W., Chen, C., Zhang, L.-j., Wang, C., & Bu, J.-j. (2010). Online detection of bursty events and their evolution in news streams. *Journal of Zhejiang University, 11*, 340–355.
Choi, J., & Croft, W. B. (2012). Temporal models for microblogs. In *CIKM '12* (pp. 2491–2494). New York, NY, USA: ACM.
Choi, J., Croft, W. B., & Kim, J. Y. (2012). Quality models for microblog retrieval. In *CIKM '12* (pp. 1834–1838). New York, NY, USA: ACM.
Crestani, F., & Markov, I. (2013). Distributed information retrieval and applications. In *ECIR* (pp. 865–868). Berlin, Heidelberg: Springer.
Croft, W. B. (2000). *Advances in information retrieval: Recent research from the center for intelligent information retrieval*. Kluwer.
Dakka, W., Gravano, L., & Ipeirotis, P. (2012). Answering general time-sensitive queries. *IEEE Transactions on Knowledge and Data Engineering, 24*(2), 220–235.
Deloule, F., Lambert, P., Beauchene, D., & Ionescu, B. (2007). Data fusion for the management of multimedia documents. In *10th International conference on information fusion, 2007* (pp. 1–7).
Diaz, F. (2005). Regularizing ad hoc retrieval scores. In *CIKM'05* (pp. 672–679).
Diaz, F. (2007). Regularizing query-based retrieval scores. *Information Retrieval, 10*(6), 531–562.
Dong, X. L., & Srivastava, D. (2013). Compact explanation of data fusion decisions, In *WWW'13* (pp. 379–390).
Donmez, P., Svore, K. M., & Burges, C. J. C. (2008). *On the local optimality of lambdarank. Microsoft research technical report*.
Duan, Y., Jiang, L., Qin, T., Zhou, M., & Shum, H. (2010). An empirical study on learning to rank tweets. In *COLING '10* (pp. 295–303).
Dwork, C., Kumar, R., Naor, M., & Sivakumar, D. (2001). Rank aggregation methods for the web. In *WWW'01* (pp. 613–622).
Efron, M. (2010). Hashtag retrieval in a microblogging environment. In *SIGIR'10* (pp. 787–788).
Efron, M. (2011). Information search and retrieval in microblogs. *Journal of the American Society for Information Science and Technology, 62*(6), 996–1008.
Erp, M. v., & Schomaker, L. (2000). Variants of the borda count method for combining ranked classifier hypotheses. In *Proceedings of the 7th international workshop on frontiers in handwriting recognition* (pp. 443–452).
Fagin, R., Kumar, R., & Sivakumar, D. (2003). Efficient similarity search and classification via rank aggregation. In *SIGMOD '03* (pp. 301–312). New York, NY, USA: ACM.
Farah, M., & Vanderpooten, D. (2007). An outranking approach for rank aggregation in information retrieval. In *SIGIR'07* (pp. 591–598).
Han, Z., Li, X., Yang, M., Qi, H., Li, S., & Zhao, T. (2012). Hit at TREC 2012 microblog track. In *TREC '12 working notes*.
He, D., & Wu, D. (2008). Toward a robust data fusion for document retrieval. In *IEEE NLP-KE'08* (pp. 1–8).
He, C., Hong, D., & Si, L. (2011). A weighted curve fitting method for result merging in federated search. In *SIGIR '11* (pp. 1177–1178). New York, NY, USA: ACM.
Hong, D., & Si, L. (2012). Mixture model with multiple centralized retrieval algorithms for result merging in federated search. In *SIGIR '12* (pp. 821–830). New York, NY, USA: ACM.
Hong, D., & Si, L. (2013). Search result diversification in resource selection for federated search. In *SIGIR '13* (pp. 613–622). New York, NY, USA: ACM.
Hoonlor, A., Szymanski, B. K., Zaki, M. J., & Chaoji, V. (2012). Document clustering with bursty. *Computing and Informatics, 31*, 1533–1555.
Horn, C., Pimas, O., Granitzer, M., & Lex, E. (2011). Realtime ad hoc search in twitter: Know-center at TREC microblog track 2011. In *Proceedings of the text retrieval conference*.
Jabeur, L. B., Damak, F., Tamine, L., Pinel-Sauvagnat, K., Cabanac, G., & Boughanem, M. (2012). IRIT at TREC microblog 2012: Adhoc task. In *TREC'12*.
Khalaman, S., & Kurland, O. (2012). Utilizing inter-document similarities in federated search. In *SIGIR'12* (pp. 1169–1170).
Kim, Y., Yeniterzi, R., & Callan, J. (2012). Overcoming vocabulary limitations in twitter microblogs. In *TREC'12*.
Klementiev, A., Roth, D., & Small, K. (2008). Unsupervised rank aggregation with distance-based models. In *ICML'08* (pp. 472–479).
Kozorovitsky, A.K., & Kurland, O. (2011). Cluster-based fusion of retrieved lists. In *SIGIR'11* (pp. 893–902).
Kurland, O., & Lee, L. (2004). Corpus structure, language models, and ad hoc information retrieval. In *SIGIR'04* (pp. 194–201).
Kustra, R., & Zagdanski, A. (2010). Data-fusion in clustering microarray data: Balancing discovery and interpretability. *IEEE/ACM Transactions on Computational Biology and Bioinformatics, 7*(1), 50–63.
Lappas, T., Arai, B., Platakis, M., Kotsakos, D., & Gunopulos, D. (2009). On burstiness-aware search for document sequences. In *SIGKDD'09* (pp. 477–486).
Lee, J. H. (1995). Combining multiple evidence from different properties of weighting schemes. In *SIGIR'95* (pp. 180–188).
Lee, G., Seo, J., Lee, S., Jung, H., Cho, B., Lee C., et al. (2001). SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *TREC'01*.
Lefortier, D., Serdyukov, P., & de Rijke, M. (2014). Online exploration for detecting shifts in fresh intent. In *CIKM 2014: 23rd ACM conference on information and knowledge management*. ACM.
Li, X., & Croft, W. B. (2003). Time-based language models. In *CIKM* (pp. 469–475).
Liang, S., & de Rijke, M. (2013). Finding knowledgeable groups in enterprise corpora. In *SIGIR'13* (pp. 1005–1008).
Liang, S., Ren, S., & de Rijke, M. (2014a). Fusion helps diversification. In *SIGIR '14, SIGIR '14* (pp. 303–312).
Liang, S., Ren, Z., & de Rijke, M. (2014b). Personalized search result diversification via structured learning. In *KDD '14* (pp. 751–760).
Liu, X., & Croft, W. B. (2008). Evaluating text representations for retrieval of the best group of documents. In *ECIR'08* (pp. 454–462).
Liu, Y.-T., Liu, T.-Y., Qin, T., Ma, Z.-M., & Li, H. (2007). Supervised rank aggregation. In *WWW'07* (pp. 481–489).
Loia, V., Pedrycz, W., & Senatore, S. (2007). Semantic web content analysis: A study in proximity-based collaborative clustering. *IEEE Transactions on Fuzzy Systems, 15*(6), 1294–1312.
Luo, Z., Osborne, M., Petrovic, S., & Wang, T. (2012). Improving twitter retrieval by exploiting structural information. In *Proceedings of the twenty-sixth AAAI conference on artificial intelligence* (pp. 648–654).
Macdonald, C., Ounis, I., Lin, J., Choudhury, A., & Soboroff, I. (2011). *Track guidelines*.
Markov, I., Arampatzis, A., & Crestani, F. (2012). Unsupervised linear score normalization revisited. In *SIGIR* (pp. 1161–1162). ACM.
Markov, I., Arampatzis, A., & Crestani, F. (2013a). On CORI result merging. In *ECIR* (pp. 752–755). Springer.
Markov, I., Azzopardi, L., & Crestani, F. (2013b). Reducing the uncertainty in resource selection. In *ECIR* (pp. 507–519). Berlin, Heidelberg: Springer.
Markov, I., & Crestani, F. (2014). Theoretical, qualitative and quantitative analyses of small-document approaches to resource selection. *ACM Transactions on Information Systems, 32*(2), 9:1–9:37.
Markovits, G., Shtok, A., & Kurland, O. (2012). Predicting query performance for fusion-based retrieval. In *CIKM'12* (pp. 813–822).
Massoudi, K., Tsagkias, M., de Rijke, M., & Weerkamp, W. (2011). Incorporating query expansion and quality indicators in searching microblog posts. In *ECIR '11* (pp. 362–367).
Mathioudakis, M., Bansal, N., & Koudas, N. (2010). Identifying, attributing and describing spatial bursts. In *VLDB'10* (pp. 1091–1102).
Metzler, D., & Cai, C. (2011). USC/ISI at TREC 2011: Microblog track. In *Proceedings of the text retrieval conference*.
Metzler, D., Cai, C., & Hovy, E. (2012). Structured event retrieval over microblog archives. In *NAACL: HLT* (pp. 646–655).
Miyanishi, T., Seki, K., & Uehara, K. (2013a). Combing recency and topic-dependent temporal variation for microblog search. In *ECIR'13* (pp. 331–343).

Miyanishi, T., Seki, K., & Uehara, K. (2013b). Improving pseudo-relevance feedback via tweet selection. In *CIKM '13* (pp. 439–448).
Montague, M., & Aslam, J. A. (2002). Condorcet fusion for improved retrieval. In *CIKM'02* (pp. 538–548).
Naveed, N., Gottron, T., Kunegis, J., & Che Alhadi, A. (2011). Searching microblogs: Coping with sparsity and document quality. In *CIKM'11* (pp. 183–188). ACM.
O'Connor, B., Krieger, M., & Ahn, D. (2010). TweetMotif: Exploratory search and topic summarization for Twitter. In *Proceedings of the fourth international AAAI conference on weblogs and social media* (pp. 384–385).
Ounis, I., Macdonald, C., Lin, J., & Soboroff, I. (2011). Overview of the TREC-2011 microblog track. In *TREC 2011. NIST*.
Peetz, M.-H., Meij, E., de Rijke, M., & Weerkamp, W. (2012). Adaptive temporal query modeling. In *ECIR '12* (pp. 455–458).
Qin, T., Geng, X., & Liu, T.-Y. (2010). A new probabilistic model for rank aggregation. In *NIPS'10* (pp. 1948–1956).
Ruzzo, W. L., & Tompa, M. (1999). A linear time algorithm for finding all maximal scoring subsequences. In *Int. conf. intelligent systems for molecular biology* (pp. 234–241).
Salakhutdinov, R., & Mnih, A. (2008). Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *ICML* (pp. 880–887).
Seo, J., & Croft, W. B. (2010). Geometric representations for multiple documents. In *SIGIR'10* (pp. 251–258).
Shaw, J. A., Fox, E. A., Shaw, J. A., & Fox, E. A. (1994). Combination of multiple searches. In *The second text retrieval conference (TREC-2)* (pp. 243–252).
Sheldon, D., Shokouhi, M., Szummer, M., & Craswell, N. (2011). LambdaMerge: Merging the results of query reformulations. In *WSDM '11* (pp. 795–804).
Shokouhi, M., & Si, L. (2011). Federated search. *Foundations and Trends in Information Retrieval, 5*(1), 1–102.
Si, L., Callan, J., Cetintas, S., & Yuan, H. (2008). An effective and efficient results merging strategy for multilingual information retrieval in federated search environments. *Information Retrieval, 11*(1), 1–24.
Soboroff, I., Ounis, I., Macdonald, C., & Lin, J. (2012). Overview of the TREC-2012 microblog track. In *TREC 2012. NIST*.
Tsagkias, M., de Rijke, M., & Weerkamp, W. (2011). Linking online news and social media. In *WSDM '11* (pp. 565–574).
Tsai, M.-F., Wang, Y.-T., & Chen, H.-H. (2008). A study of learning a merge model for multilingual information retrieval. In *SIGIR'08* (pp. 195–202).
Vlachos, M., Meek, C., & Vagena, Z. (2004). Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD'04* (pp. 131–142).
Weerkamp, W., & de Rijke, M. (2012). Credibility-inspired ranking for blog post retrieval. *Information Retrieval Journal, 15*(3–4), 243–277.
Wei, Z., Gao, W., Zhou, L., Li, B., & Wong, K.-F. (2011). Exploring tweets normalization and query time sensitivity for twitter search. In *Proceedings of the text retrieval conference*.
Wei, B., Zhang, S., Li, R., & Wang, B. (2012). A time-aware language model for microblog retrieval. In *TREC'12*.
Wu, S. (2012). *Data fusion in information retrieval. Adaptation, Learning and Optimization* (Vol. 13). Springer.
Yang, J., & Leskovec, J. (2011). Patterns of temporal variation in online media. In *WSDM '11* (pp. 177–186). ACM.
Zhang, X., Hui, K., He, B., & Luo, T. (2011). Gucas at TREC-2011 microblog track. In *TREC'11*.
Zhao, D., & Rosson, M. B. (2009). How and why people twitter: the role that micro-blogging plays in informal communication at work. In *GROUP '09* (pp. 243–252).
Zhu, Y., & Shasha, D. (2003). Efficient elastic burst detection in data streams. In *SIGKDD'03* (pp. 336–345).