Reproducible Experiments on Lexical and Temporal Feedback for Tweet Search

Jinfeng Rao[1], Jimmy Lin[1], and Miles Efron[2]

[1] University of Maryland, College Park
jinfeng@cs.umd.edu, jimmylin@umd.edu
[2] University of Illinois, Urbana-Champaign
mefron@illinois.edu

Abstract. "Evaluation as a service" (EaaS) is a new methodology for community-wide evaluations where an API provides the only point of access to the collection for completing the evaluation task. Two important advantages of this model are that it enables reproducible IR experiments and encourages sharing of pluggable open-source components. In this paper, we illustrate both advantages by providing open-source implementations of lexical and temporal feedback techniques for tweet search built on the TREC Microblog API. For the most part, we are able to reproduce results reported in previous papers and confirm their general findings. However, experiments on new test collections and additional analyses provide a more nuanced look at the results and highlight issues not discussed in previous studies, particularly the large variances in effectiveness associated with training/test splits.

Keywords: TREC Microblog, evaluation as a service, search API.

1 Introduction

"Evaluation as a service" (EaaS) [11] is a new methodology that enables community-wide evaluations and the construction of test collections on documents that cannot be distributed. The basic idea is that instead of providing the document collection in a downloadable form, as is standard in most TREC, NTCIR, CLEF, and other evaluations, the organizers provide a service API through which the evaluation task can be completed [12]. Typically, the API would provide keyword search capabilities, but it can be augmented with additional features customized to the evaluation task at hand. The key point is, however, that the API provides the sole access point to the document collection, and thus it can be engineered to respect restrictions on the dissemination of content.

One important advantage of the evaluation-as-a-service model is that it enables reproducible IR experiments. Modern search systems have become complex collections of components for document ingestion, inverted indexing, query evaluation, document ranking, and machine learning. As a result, it can be difficult to isolate and attribute differences in effectiveness to specific components, algorithms, or techniques. Consider a baseline retrieval model such as BM25 or

A. Hanbury et al. (Eds.): ECIR 2015, LNCS 9022, pp. 755–767, 2015.

query-likelihood within the language modeling framework—alternative implementations may produce substantially different results due to small but consequential decisions such as the tokenization strategy, stemming algorithm, method for pruning the term space (e.g., discarding long or rare terms), minor scoring variations, and other engineering issues [16,22]. In some cases, the effects that we are hoping to study are masked by differences we are not interested in. The evaluation-as-a-service model addresses many of these issues by deploying a common API that is used by all participants. This means that everything "below" the API (e.g., indexing, tokenization, etc.) is *exactly* the same for everyone. Thus, we can be confident that differences in effectiveness can be attributed to retrieval techniques on top of the API, rather than "uninteresting" issues.

Additionally, we believe that this model is conducive to an open culture of sharing pluggable system components. There is broad recognition that open-source software advances the state of the art; a common API increases the likelihood that code components inter-operate, thus increasing the likelihood of adoption. Although there is already widespread availability of open-source search engines, nearly all systems are monolithic in that they were not designed for service decomposition along functional boundaries. This means that a particular algorithm developed for one system cannot be easily used by researchers who have written their code on another system due to interface incompatibilities. A common API begins to address these issues.

In this paper, we illustrate both advantages of the evaluation-as-a-service model by reproducing lexical and temporal feedback techniques for searching tweets in the context of the TREC Microblog tracks, which was the first to operationalize this evaluation model. By lexical feedback we mean pseudo-relevance feedback where an initial set of retrieved documents is exploited to refine the query model. Since tweets are very short, a number of researchers have suggested that the query expansion effects of pseudo-relevance feedback are beneficial to search effectiveness. To rigorously test this insight, we have reimplemented the popular RM3 approach [8] using the TREC Microblog API. We use the term *lexical* feedback to distinguish RM3 (and related models) from techniques that take advantage of the *temporal* information of documents, which is the focus of our second set of experiments. We attempt to reproduce the techniques proposed in a recent SIGIR paper by Efron et al. [5], reimplementing their proposed algorithms based on kernel density estimation (KDE) as well as two other temporal-ranking techniques. Finally, we combine both lexical and temporal feedback to explore the question of whether the effectiveness gains are cumulative. All of the source code for experiments conducted in this paper can be found in our open-source code repository.[1]

Our reproducibility efforts were largely successful and our experimental results are consistent with previous studies for the most part. However, through more extensive experiments on new test collections and additional analyses, we provide a more nuanced look at previous results. In particular, we note the large variances in effectiveness associated with training/test splits of test collections.

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

2 Background

The context for our study is the recent Microblog tracks at TREC [17,21,11], which have been running since 2011. Although the task has remained essentially the same, the evaluation methodology has changed over the years, and so it is worth providing an overview.

The TREC Microblog tracks in 2013 and 2014 used the evaluation-as-a-service model described in the introduction. The API served the Tweets2013 collection, which consists of 243 million tweets crawled from Twitter's public sample stream between February 1 and March 31, 2013 (inclusive). Although the "official" collection is not available for download, participants could acquire substantively similar data by also crawling the public stream during the same time (which was coordinated on the track mailing list and indeed, many participants did acquire tweets in this manner). In contrast, the 2011 and 2012 evaluations used the Tweets2011 corpus, which consists of an approximately 1% sample (after some spam removal) of tweets from January 23, 2011 to February 7, 2011 (inclusive), totaling approximately 16 million tweets. For those evaluations, the TREC organizers made the list of tweet ids that comprise the collection available, and together with a distributed crawler (also supplied by the organizers), participants could download the actual tweets from Twitter itself (this approach does not scale to the much larger Tweets2013 collection). Note, however, that the collection acquired by each participant might be slightly different due to transient network glitches, message deletions, removal of spam accounts, and a whole host of other factors. Nevertheless, a study in 2012 [13] found that these artifacts did not impact the stability of the test collection. These methodological differences add an extra dimension of interest in our studies, as we would like to examine the impact of having a local collection vs. using the service API.

The formulation of the tweet search problem for the TREC Microblog track is as follows: at time t, a user expresses an information need in the form of a query Q. The system's task is to return topically-relevant documents (tweets) posted before the query time. Thus, each topic consists of a query and an associated timestamp, which indicates when the query was issued. There are 50 topics for TREC 2011, 60 topics for TREC 2012, 60 topics for TREC 2013, and 55 topics for TREC 2014. NIST assessors used a standard pooling strategy for evaluation, assigning one of three judgments to each tweet in the pool: "not relevant", "relevant", and "highly relevant". For the purpose of our experiments, we considered both "relevant" and "highly relevant" tweets to be relevant.

In addition to the official API used for TREC 2013 and 2014 (which served the Tweets2013 collection), the organizers also provided an API that serves the smaller Tweets2011 collection so that participants could runs experiments using topics from TREC 2011 and 2012. Both APIs were identical except for the underlying document collection, and were implemented in Java using service definitions provided by Thrift[2] and Lucene[3] as the underlying search engine.

[2] `http://thrift.apache.org/`
[3] `http://lucene.apache.org/`

Ranking was provided using Lucene's implementation of query-likelihood in the language modeling framework [18]. The API returned up to 10000 hits, and each hit contained the full text of the tweet and associated metadata (statistics about the user, the source tweet if the tweet was a retweet or a reply, etc.). There is one implementation detail worth mentioning—for efficiency reasons, Lucene implements a rank-equivalent scoring model to query-likelihood,[4] which cannot be used in more complex ranking models that depend on valid log probabilities. To get around this issue, a patch was made to the service API whereby the client could (optionally) request that the system compute valid query-likelihood probabilities in a second pass after the initial retrieval. In all our experiments, we enabled this option.

Code for all experiments reported in this paper, implemented in Java, has been open sourced and integrates directly with the TREC Microblog API. Reproducing our results is as simple as executing the command-line invocations included in our documentation—the evaluation-as-a-service model obviates the need to download the document collection, build inverted indexes, etc.

3 Lexical Feedback with Relevance Models

A longstanding challenge in information retrieval is the issue of vocabulary mismatch, where queries are expressed using terms not present in relevant documents. Query expansion techniques, particularly those based on pseudo-relevance feedback, are often used to address this problem; there is a long history of research in this area dating back many decades [19]. The brevity of tweets exacerbates vocabulary mismatch, and thus query expansion techniques are likely to improve the effectiveness of tweet search. This is an insight shared by many researchers—for example, many of the most effective runs from TREC 2011 take advantage of such techniques in various guises [2,10,14]. In our first set of experiments, we wished to verify the effectiveness of pseudo-relevance feedback for tweet search, and to that end, we implemented relevance models [8] using the TREC Microblog API. Relevance models have specifically been explored for tweet search in a few previous studies [3,15], which provides a point of reference for our reproducibility efforts.

Given a query Q consisting of n query terms $\{q_1, q_2, ...q_n\}$, its relevance model $P(w|R_Q)$ is simply a weighted average of the terms in all documents, where the weights are the query likelihood scores:

$$P(w|R_Q) = \sum_{D \in \mathcal{D}} P(D)P(w|D) \prod_{i=1}^{n} P(q_i|D). \quad (1)$$

In the RM3 variant [1], the above model is interpolated with the observed query model according to a mixing parameter γ. In our experiments, we set $\gamma = 0.5$, which is the default value in the Indri implementation. In practice, RM3 is typically implemented using query expansion (e.g., augmenting the original query

[4] See equation (4) in [24] for more details.

Table 1. Results comparing query-likelihood (QL) against RM3, where the relevance models are estimated with retweets included (+retweets) or discarded (−retweets)

	2011/12		2013/14	
Method	MAP	P30	MAP	P30
QL	0.2692	0.3552	0.3266	0.5156
RM3 (+retweets)	0.3005*	0.3778*	0.3629*	0.5351*
RM3 (−retweets)	0.3003*	0.3787*	0.3597*	0.5357*

using Indri query operators); our implementation follows this approach as well. Following common parameter settings, we estimated the relevance models from $k = 50$ pseudo-relevant documents and selected $n = 20$ feedback terms.

Experimental results are shown in Table 1 for TREC 2011/12 and 2013/14, reporting mean average precision (MAP) to 1000 hits and precision at rank 30 (P30) computed with `trec_eval`. There is no training/test split because we are not tuning parameters, but simply using "best practice" defaults from the literature. Query-likelihood provides a baseline for comparison. The symbol * indicates that the difference with respect to the baseline is statistically significant ($p < 0.01$) based on Fisher's two-sided, paired randomization test [20]. Our experiments also examined the impact of one detail we have not seen much discussion about in the literature: the effect of retweets. According to the assessment guidelines, retweets that provide no additional information are considered not relevant. Thus, it makes sense to remove all retweets from the final results (which we did here and for all subsequent runs). However, it is unclear if the retweets should be included or discarded when estimating relevance models—thus, we tried both conditions.

Results show that RM3 yields significant and consistent improvements over the query-likelihood baseline in terms of MAP and P30. Furthermore, it does not appear to matter whether or not retweets are included in the estimation of the relevance model. To summarize, we have successfully reproduced previous results and confirmed that the benefits of pseudo-relevance feedback are robust.

4 Temporal Feedback with Kernel Density Estimation

4.1 Overview

Relevance models represent a popular approach to lexical feedback, taking advantage of an initial set of search results to refine the system's estimate of the term distribution of relevant documents. We can extend this idea to temporal feedback by estimating the temporal density of relevance—which characterizes where along a timeline we would expect relevant documents to appear. For information needs where temporality plays an important factor (as is common in tweet search), we would expect a non-uniform distribution of documents over time, and hence there might be a temporal relevance signal that can be exploited. In the same way that an initial set of search results can be used to estimate relevance models, we can estimate the temporal density of relevance

from an initial list of retrieved documents. These are the ideas behind the work of Efron et al. [5], which we reproduce here. Below, we briefly summarize the relevant techniques.

As a starting point, consider the query-likelihood approach in the language modeling framework [18]. Documents are ranked by $P(D|Q) \propto P(Q|D)P(D)$, where $P(Q|D)$ is the likelihood that the language model that generated document D would also generate query Q, and $P(D)$ is the prior distribution.

Recency Priors. Li and Croft [9] incorporate temporal information using a prior that favors recent documents, modeling $P(D)$ with an exponential $P(D) = \lambda e^{-\lambda T_D}$, where T_D is the timestamp of document D and $\lambda \geq 0$ is the rate parameter. We refer to this as a *recency prior*, or "Recency" for short.

Moving Window Approach. Recency priors are query-independent and unable to account for information needs with different temporal profiles [7]. Dakka et al. [4] proposed a query-specific way to combine lexical and temporal evidence in the language modeling framework by separating the lexical and temporal signals into two components: W_D, the words in the document and T_D, the document's timestamp. This leads to the following derivation:

$$P(D|Q) = P(W_D, T_D|Q) \tag{2}$$
$$= P(T_D|W_D, Q)P(W_D|Q) \tag{3}$$
$$\sim P(W_D|Q)P(T_D|Q) \tag{4}$$

where the last step follows if we assume independence between lexical and temporal evidence. The result is similar to standard query-likelihood, but with the addition of the probability of observing a time T_D given the query Q.

Dakka et al. proposed several ways to estimate $P(T_D|Q)$. In the moving window approach (WIN for short), initial documents retrieved for Q are allocated among b bins according to their timestamps. For each bin b_t, we count $n(b_t)$, the number of retrieved documents in b_t. Next, bin counts are smoothed by averaging x bins into the past and x bins into the future. Let $n(b_{tx})$ be the average number of documents in the $2x$ bins surrounding b_t and b_t itself. Finally, bins are arranged in decreasing order of $n(b_{tx})$. The quantity $P(T_D|Q)$ depends on the bin associated with T_D. If T_D is in the n^{th} ordered bin, then $P(T_D|Q) = \phi(n, \lambda)$ where ϕ is an exponential distribution with rate parameter λ.

Kernel Density Estimates. To estimate the temporal distribution of relevant documents, Efron et al. proposed using kernel density estimation (KDE) [6]. Let $\{x_1, x_2, \ldots, x_n\}$ be i.i.d. samples drawn from some distribution with an unknown density f. We are interested in estimating the shape of this function f. Its kernel density estimator is:

$$\hat{f}_\omega(x) = \frac{1}{nh} \sum_{i=0}^{n} \omega_i K\left(\frac{x - x_i}{h}\right) \tag{5}$$

where $K(\cdot)$ is the kernel—a symmetric function that integrates to one (in our case, a Gaussian)—and $h > 0$ is a smoothing parameter called the bandwidth.

For bandwidth selection we use what is known as the "robust rule of thumb" [23], which yields a bandwidth automatically. KDE additionally has the ability to handle weighted observations, given non-negative weights $\{\omega_1, \omega_2, \ldots, \omega_n\}$ such that $\sum \omega_i = 1$. Consider four different weighting schemes:

- *Uniform weights.* The simplest approach is to give all documents in the initial results equal weights.
- *Score-based weights.* We can weight each document based on its query-likelihood, i.e.,

$$\omega_i^s = \frac{P(Q|D_i)}{\sum_{j=1}^{n} P(Q|D_j)}. \tag{6}$$

- *Rank-based weights.* We can adopt a rank-based scheme that preserves ordering in the initial results, but not the actual score differences, via an exponential distribution:

$$\omega_i^r = \frac{\lambda e^{-\lambda r_i}}{\sum_{j=1}^{n} \lambda e^{-\lambda r_i}} \tag{7}$$

  where $\lambda > 0$ is the rate parameter of the exponential and r_i is the rank of document D_i in R. Though we could leave λ as a tuneable parameter, a simpler approach is to use the maximum likelihood estimate. If R contains n documents, the MLE of λ is simply $\frac{1}{\bar{r}}$, where $\bar{r}$ is the mean of the ranks $1, 2, \ldots, n$.
- *Oracle.* An upper bound can be characterized by an oracle where the density estimates are derived from documents marked relevant by human assessors.

To combine the temporal and lexical evidence, Efron et al. proposed a simple log-linear model. For a parameter $\alpha \in [0, 1]$, we have

$$\log P_\alpha(R|D, Q) = Z_\alpha + (1 - \alpha) \log P(R|W_D, Q) + \alpha \log P(R|T_D, Q) \tag{8}$$

where Z_α is a normalization constant. Since Z_α does not depend on D for ranking, we can ignore it; α is a free parameter learned from data.

4.2 Experimental Results

Experiments in Efron et al. were conducted on topics from TREC 2011 and 2012 over a local copy of the Tweets2011 corpus. During corpus preparation, all retweets were eliminated. Thus, the collection used in those experiments is substantively different from the corpus behind the official TREC Microblog API, which does include retweets. This is an additional factor that might affect the reproducibility of their results. To be clear, however, retweets are removed in all cases in the final ranked list prior to evaluation.

In our first set of experiments, we attempted to reproduce the experimental conditions in Efron et al. as closely as possible. Even-numbered topics from TREC 2011 and 2012 were used for training, and odd-numbered topics for

Table 2. Results from attempting to reproduce experiments in Efron et al. [5] as closely as possible. Metrics computed over odd-numbered topics from TREC 2011/12, training on even-numbered topics. Columns marked "original" contain results copied from the previous SIGIR paper; columns marked "reproduced" show reproduced results.

	MAP		**P30**	
Condition	**original**	**reproduced**	**original**	**reproduced**
QL	0.2363	0.2705	0.3473	0.3582
Recency	0.2467°	0.2766	0.3642°	0.3607
WIN	0.2407	0.2548	0.3515	0.3449
KDE (uniform)	0.2457°	0.2685	0.3618°	0.3534
KDE (score-based)	0.2505$^{\bullet\dagger}$	0.2719	0.3606°	0.3582
KDE (rank-based)	0.2546$^{\bullet\triangle\dagger}$	0.2724	0.3709$^{\bullet\ddagger}$	0.3649
KDE (oracle)	0.2843$^{\bullet\blacktriangle\ddagger}$	0.3045$^{\bullet\blacktriangle\ddagger}$	0.4024$^{\bullet\blacktriangle\ddagger}$	0.3922$^{\bullet\blacktriangle\ddagger}$

Table 3. Symbols indicating statistically significant change for data reporting

Symbol	Description
$\circ$, $\bullet$	improvements over the QL baseline ($p < 0.05$, $p < 0.01$)
$\triangle$, $\blacktriangle$	improvements over the recency prior ($p < 0.05$, $p < 0.01$)
†, ‡	improvements over the WIN method ($p < 0.05$, $p < 0.01$)

testing. These results are shown in Table 2, with QL representing the query-likelihood baseline; we characterize effectiveness in terms of MAP (to rank 1000) and precision at rank 30 (P30). The free parameters for each technique were tuned via grid search to optimize MAP and P30 (separately).[5] Results are annotated with symbols indicating the statistical significance of improvements as shown in Table 3. Following Efron et al., we applied one-sided paired t-tests for significance testing.

These results are somewhat different from those reported in Efron et al. We immediately noticed the differences between the two versions of the QL baseline (Indri for the SIGIR paper and Lucene + QL recomputation for the TREC Microblog API). Although both are putatively implementing the same ranking function (Dirichlet scores), there is a fairly large difference in MAP. There are many possible sources for these differences, including the fact that the two experiments are actually on *different* collections, as well as issues related to corpus preparation such as removal of retweets, stemming, tokenization, etc. This further affirms the arguments behind the evaluation-as-a-service model in providing a common starting point for everyone—otherwise, relatively uninteresting differences could easily mask the effects of the techniques we are studying.

Consistent with the original work, we find that the KDE oracle condition is highly effective, which indicates that there is a strong temporal relevance signal. We observe improvements for rank-based KDE in terms of MAP and P30, albeit

[5] Note that in these experiments we did not compute metrics using `trec_eval` to facilitate tighter training/test coupling, and thus there are small differences between our values and those reported using `trec_eval` due to how scoring ties are handled.

Table 4. Results from TREC 2011/12. "Cross" represents training using all TREC 2013/14 topics; "Even-Odd" represents training on even-numbered topics and testing on odd-numbered topics; "Odd-Even" represents switching train/test.

	Cross		**Even-Odd**		**Odd-Even**	
Metric	**MAP**	**P30**	**MAP**	**P30**	**MAP**	**P30**
QL	0.2689	0.3562	0.2705	0.3582	0.2673	0.3541
Recency	0.2748	0.3578	0.2766	0.3607	0.2721	0.3509
WIN	0.2689	0.3578	0.2548	0.3449	0.2673	0.3541
KDE (uniform)	0.2699	0.3568	0.2685	0.3534	0.2702	0.3553
KDE (score-based)	0.2711	0.3673	0.2719	0.3582	0.2697	0.3698
KDE (rank-based)	0.2707	0.3655	0.2724	0.3649	0.2716	0.3616
KDE (oracle)	$0.3032^{\bullet\blacktriangle\ddagger}$	$0.3988^{\bullet\blacktriangle\ddagger}$	$0.3045^{\bullet\blacktriangle\ddagger}$	$0.3922^{\bullet\blacktriangle\ddagger}$	$0.3001^{\bullet\blacktriangle\ddagger}$	$0.4069^{\bullet\blacktriangle\dagger}$

not statistically significant. Furthermore, the WIN approach does not seem to be effective. We suspect that these findings may be attributed to the much improved QL baseline in our experiments.

We extended the experiments of Efron et al. in two ways. First, we evaluated the techniques on test collections from TREC 2013 and 2014. This not only provides (roughly) double the number of topics, but also allows us to examine the effects of a much larger corpus. An open question is whether the proposed techniques would remain effective for a collection spanning a much longer duration (about two weeks for Tweets2011 compared to two months for Tweets2013); we now have an opportunity to answer this question. Second, during our experiments we noticed large effectiveness differences that stemmed from different training/test splits; we wanted to explore these effects in more detail.

In terms of training regimes, one simple approach is to arbitrarily divide the test collection into halves; train on one half and test on the other half. Splitting topics by topic number is a perfectly acceptable arbitrary division: we can train on even-numbered topics and test on odd-numbered topics (as before), and also flip the two halves (i.e., train on odd, test on even). Another reasonable strategy might be to consider the TREC 2011/12 topics to be a unit, train on all those topics, and test on TREC 2013/14 topics; and also the other way around, i.e., train on TREC 2013/14 topics and test on all TREC 2011/12 topics. This condition assesses whether it may be possible to generalize parameters across different collections, i.e., a simple form of transfer learning.

Results from these experiments are shown in Table 4 for TREC 2011/12 and Table 5 for TREC 2013/14. Note that the figures reported in Table 2 are the same as those in the "Even-Odd" column in Table 4. In these experiments we used Fisher's two-sided, paired randomization test [20], reflecting better practice than the one-sided paired t-tests used in the SIGIR experiments. Results appear to show that the KDE techniques are more effective on the TREC 2013/2014 test collection. For rank-based weights, the differences are statistically significant in most cases.

To further explore the train/test split issue, we conducted a series of trials where we randomly divided the TREC 2011/12 and TREC 2013/14 topics in

Table 5. Results from TREC 2013/14. "Cross" represents training using all TREC 2011/12 topics; other conditions have the same meaning as in Table 4.

	Cross		Even-Odd		Odd-Even	
Metric	**MAP**	**P30**	**MAP**	**P30**	**MAP**	**P30**
QL	0.3139	0.5197	0.3559	0.5638	0.2712	0.4749
Recency	0.3129	0.5336	0.3593	0.5736	0.2773	0.4994
WIN	0.3139	0.5197	0.3559	0.5638	0.2712	0.4749
KDE (uniform)	0.3121	0.5177	0.3490	0.5603	0.2737	0.4795
KDE (score-based)	0.3140	0.5206	0.3516	0.5747	0.2753	0.4795
KDE (rank-based)	0.3267 $^{\bullet\ddagger}$	0.5542$^{\bullet\ddagger}$	0.3600	0.5983$^{\bullet\blacktriangle\ddagger}$	0.2949 $^{\bullet\triangle\ddagger}$	0.5228 $^{\bullet\ddagger}$
KDE (oracle)	0.3492$^{\bullet\blacktriangle\ddagger}$	0.5829$^{\bullet\blacktriangle\ddagger}$	0.3816$^{\bullet\blacktriangle\dagger}$	0.6328$^{\bullet\blacktriangle\ddagger}$	0.3135$^{\bullet\blacktriangle\ddagger}$	0.5363$^{\bullet\blacktriangle\ddagger}$

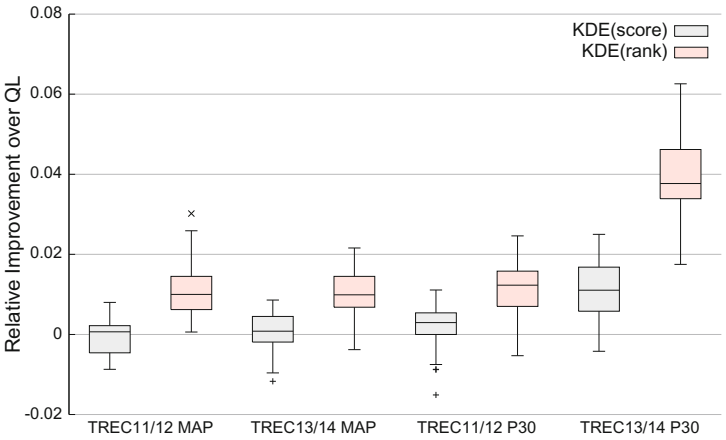

Fig. 1. Box-and-whiskers plots summarizing effectiveness differences (with respect to QL baseline) of score-based and rank-based weights for KDE, across 50 random trials where the topics are split in half for training/test.

half. For each trial, we trained on half the topics and tested on the other half. We then computed the effectiveness differences between each technique and the QL baseline. These differences, collected over 50 trials, are summarized in box-and-whiskers plots in Figure 1 for KDE with score-based weights and KDE with rank-based weights. We show the distribution of effectiveness differences in terms of MAP and P30 on TREC 2011/12 and TREC 2013/14. Following convention: Each box represents the span between the first and third quartiles, with a horizontal line at the median value. Whiskers extend from the ends of each box to the most distant point whose value lies within 1.5 times the interquartile range. Points that lie outside these limits are drawn individually. These results capture the overall effectiveness of the techniques better than metrics from any single arbitrary split. Here, we clearly see that rank-based weights are more effective than score-based weights.

Table 6. Results from attempting to reproduce lexical+temporal feedback experiments in Efron et al. [5] as closely as possible. Metrics computed over odd-numbered topics from TREC 2011/12.

	MAP		P30	
Condition	**original**	**reproduced**	**original**	**reproduced**
RM3	0.2897	0.2847	0.3843	0.3684
KDE (score-based)	0.3014*	0.2834	0.4079*	0.3570
KDE (rank-based)		0.2703		0.3509*
KDE (oracle)		0.3027*		0.3945*

Taken as a whole, our findings are mostly consistent with the results of Efron et al. We find that KDE with rank-based weights yields improvements over the QL baseline, which affirms the overall effectiveness of the proposed approach.

5 Combining Lexical and Temporal Feedback

The final question explored in Efron et al. was whether lexical relevance signals are distinct from temporal relevance signals—in practical terms, are the effectiveness gains from both techniques additive?

Results from applying KDE on top of an RM3 baseline are shown in Table 6, reproducing the experiments in Efron et al. as closely as possible. In this case, we used the parameter setting that optimized MAP from the experiments in Table 2. The original SIGIR paper reported only score-based weights for KDE, but here we include the rank-based weights and the oracle condition as well. In this and the following experiments, relevance models were estimated with retweets included. The symbol * indicates that the difference with respect to the RM3 baseline is statistically significant ($p < 0.05$). We find that the oracle condition is significantly better than the RM3 baseline for both metrics; rank-based weighting for P30 is significantly worse, but none of the other differences are significant. This is not consistent with the findings in Efron et al., who reported significant improvements for score-based weights.

To further examine these inconsistencies, we repeated the experiments on all topics from TREC 2011/12 and TREC 2013/14. Here, we used the parameter settings in the "cross" condition that optimizes MAP from Tables 4 and 5. The symbol * indicates that the difference with respect to RM3 is statistically significant ($p < 0.05$). We see that neither score-based nor rank-based KDE is able to improve upon RM3, although the oracle condition shows a significant improvement in all cases. This confirms that a temporal relevance signal exists independently of the lexical relevance signal, although it does not appear that the proposed non-oracle techniques can exploit this signal. Note that in these experiments, we simply used previous parameters; perhaps with retuning we might more closely replicate previous results.

Table 7. Applying temporal feedback on top of lexical feedback for TREC 2011/12 and TREC 2013/14 data

	2011/12		2013/14	
Method	MAP	P30	MAP	P30
RM3	0.3005	0.3778	0.3629	0.5351
KDE (score-based)	0.2925*	0.3781	0.3543	0.5279
KDE (rank-based)	0.2769*	0.3642	0.3670	0.5423
KDE (oracle)	0.3197*	0.4191*	0.3964*	0.5952*

6 Conclusions

In this paper, we have successfully reproduced experiments described in previous work involving lexical feedback and temporal feedback. A third set of experiments on the combination of lexical and temporal feedback met with limited success. More in-depth analyses and extensions to new test collections add more nuance to previous conclusions.

We feel that there are two important takeaway lessons: First, though it may seem like an obvious point, meaningful comparisons depend on a proper baseline. We suspect that improvements reported in previous studies disappeared or were diminished to a large extent because the baseline became more competitive in our experiments. IR researchers must continuously remain vigilant and be "honest" with themselves in presenting a fair point of comparison.

Second, we found that effectiveness is highly dependent on the training/test split. This is perhaps not surprising since TREC test collections are relatively small—however, we see in the literature plenty of papers that base their conclusions on a single (arbitrary) training/test split. It is difficult to rule out that those findings, even in the case of statistically significant improvements, are due to fortuitous splits of the data. Running many randomized trials, as we have done in this paper, provides a more complete characterization of effectiveness differences. Perhaps such practices should become more commonplace in information retrieval evaluation.

Acknowledgments. This work was supported in part by the U.S. National Science Foundation under grants 1217279 and 1218043. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of the sponsor.

References

1. Abdul-Jaleel, N., Allan, J., Croft, W.B., Diaz, F., Larkey, L., Li, X., Metzler, D., Smucker, M.D., Strohman, T., Turtle, H., Wade, C.: UMass at TREC 2004: Novelty and HARD. In: TREC (2004)
2. Amati, G., Amodeo, G., Bianchi, M., Celi, A., Nicola, C.D., Flammini, M., Gaibisso, C., Gambosi, G., Marcone, G.: FUB, IASI-CNR, UNIVAQ at TREC 2011 Microblog track. In: TREC (2011)

3. Choi, J., Croft, W.B.: Temporal models for microblog. In: CIKM, pp. 2491–2494 (2012)
4. Dakka, W., Gravano, L., Ipeirotis, P.G.: Answering general time-sensitive queries. IEEE Transactions on Knowledge and Data Engineering 24(2), 220–235 (2012)
5. Efron, M., Lin, J., He, J., de Vries, A.: Temporal feedback for tweet search with non-parametric density estimation. In: SIGIR, pp. 33–42 (2014)
6. Hall, P., Turlach, B.A.: Reducing bias in curve estimation by use of weights. Computational Statistics & Data Analysis 30(1), 67–86 (1999)
7. Jones, R., Diaz, F.: Temporal profiles of queries. ACM Transactions on Information Systems 25(3), Article 14 (2007)
8. Lavrenko, V., Croft, W.B.: Relevance based language models. In: SIGIR, pp. 120–127 (2001)
9. Li, X., Croft, W.B.: Time-based language models. In: CIKM, pp. 469–475 (2003)
10. Li, Y., Zhang, Z., Lv, W., Xie, Q., Lin, Y., Xu, R., Xu, W., Chen, G., Guo, J.: PRIS at TREC2011 Micro-blog track. In: TREC (2011)
11. Lin, J., Efron, M.: Overview of the TREC-2013 Microblog Track. In: TREC (2013)
12. Lin, J., Efron, M.: Infrastructure support for evaluation as a service. In: WWW Companion, pp. 79–82 (2014)
13. McCreadie, R., Soboroff, I., Lin, J., Macdonald, C., Ounis, I., McCullough, D.: On building a reusable Twitter corpus. In: SIGIR, pp. 1113–1114 (2012)
14. Metzler, D., Cai, C.: USC/ISI at TREC 2011: Microblog track. In: TREC (2011)
15. Miyanishi, T., Seki, K., Uehara, K.: Improving pseudo-relevance feedback via tweet selection. In: CIKM, pp. 439–448 (2013)
16. Mühleisen, H., Samar, T., Lin, J., de Vries, A.: Old dogs are great at new tricks: Column stores for IR prototyping. In: SIGIR, pp. 863–866 (2014)
17. Ounis, I., Macdonald, C., Lin, J., Soboroff, I.: Overview of the TREC-2011 Microblog Track. In: TREC (2011)
18. Ponte, J.M., Croft, W.: A language modeling approach to information retrieval. In: SIGIR, pp. 275–281 (1998)
19. Rocchio, J.J.: Relevance feedback in information retrieval. In: Salton, G. (ed.) The SMART Retrieval System—Experiments in Automatic Document Processing, pp. 313–323. Prentice-Hall, Englewood Cliffs, New Jersey (1971)
20. Smucker, M.D., Allan, J., Carterette, B.: A comparison of statistical significance tests for information retrieval evaluation. In: CIKM, pp. 623–632 (2007)
21. Soboroff, I., McCullough, D., Lin, J., Macdonald, C., Ounis, I., McCreadie, R.: Evaluating real-time search over tweets. In: ICWSM, pp. 579–582 (2012)
22. Trotman, A., Puurula, A., Burgess, B.: Improvements to BM25 and language models examined. In: ADCS (2014)
23. Turlach, B.A.: Bandwidth selection in kernel density estimation: A review (1993)
24. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Computing Surveys 38(6), 1–56 (2006)