Mining the Temporal Statistics of Query Terms for Searching Social Media Posts

Jinfeng Rao,[1] Ferhan Ture,[2] Xing Niu,[1] and Jimmy Lin[3]
[1] Department of Computer Science, University of Maryland
[2] Comcast Applied AI Research Lab
[3] David R. Cheriton School of Computer Science, University of Waterloo
{jinfeng,xingniu}@cs.umd.edu,ferhan_ture@cable.comcast.com,jimmylin@uwaterloo.ca

ABSTRACT

There is an emerging consensus that time is an important indicator of relevance for searching streams of social media posts. In a process similar to pseudo-relevance feedback, the distribution of document timestamps from the results of an initial query can be leveraged to infer the distribution of relevant documents, for example, using kernel density estimation. In this paper, we explore an alternative approach to mining relevance signals directly from the temporal statistics of query terms in the collection, without the need to perform an initial retrieval. We propose two approaches: a linear ranking model that combines features derived from temporal collection statistics of query terms and a regression-based method that attempts to directly predict the distribution of relevant documents from query term statistics. Experiments on standard tweet test collections show that our proposed methods significantly outperform competitive baselines. Furthermore, studies of different feature combinations show the extent to which different types of temporal signals impact retrieval effectiveness.

1 INTRODUCTION

There is a large body of literature in information retrieval that has established the importance of understanding and modeling the temporal distribution of documents as well as queries for various information seeking tasks [5–9, 12, 16]. This is particularly important for searching rapidly-evolving, real-time social media streams such as Twitter, which is the focus of this work. Given an information need expressed as a query, we wish to develop ranking models that return relevant tweets. We refer to this problem as temporal ranking to emphasize the need to model temporal aspects of the information need as well as the document collection.

One successful approach to temporal ranking is to estimate the distribution of relevant documents using the distribution of document timestamps from the results of an initial query [8]. In the same way that pseudo-relevance feedback uses the results of an initial query to refine estimates of term distributions in relevant documents, this class of techniques can be viewed as performing inference on the distribution of document timestamps. The theoretical foundation of this approach lies in the temporal cluster hypothesis [8], which is the observation that relevant documents tend to cluster together in time. One effective implementation of this idea is to use kernel density estimation (KDE) to infer a "temporal prior" for a given information need.

In this work, we take a different approach to estimate the distribution of relevant documents: instead of relying on the results of an initial query, we attempt to exploit temporal signals embedded in the distribution of the query terms themselves. We call these *query trends*, which are generalizations of collection term statistics (of query unigrams and bigrams) in the temporal dimension. Specifically, we keep track of the number of occurrences of query terms across a moving window over the document collection.

Consider an example that illustrates our intuition: the distribution of relevant documents (i.e., from human judgments) for topic MB127 ("hagel nomination filibustered") from the TREC 2013 Microblog Track is shown on the top in Figure 1. The x axis denotes a timeline, with units in days anchored at the query time on the right edge. Of course, this distribution is not known at query time—it is the target of our prediction. The remaining rows in Figure 1 show *query trends*, the distribution of query terms in the collection across time, for the unigrams "filibustered", "hagel", "nomination", and the bigram "hagel nomination". Informally, our problem can be characterized as using query trends to predict the distribution of relevant documents (i.e., the top row in Figure 1).

From this example, it is apparent that there are correlations between query trends and the distribution of relevant documents. Furthermore, a key advantage of our approach over previous methods is that it eliminates the need for an initial retrieval, since temporal collection statistics can be compactly stored for efficient lookup [20] during query processing. From an efficiency perspective, this means that models based solely on query trends can be substantially faster than those that require an initial retrieval.

In this paper, we explore two different approaches to exploiting query trends:

- A linear ranking model that combines features based on the temporal collection statistics of query unigrams and bigrams, their entropies, other related signals.
- A regression-based method that attempts to directly predict the distribution of relevant documents from unigram and bigram query trends.

These two approaches are further combined in an ensemble model, which additionally includes features derived from previous work based on kernel density estimation.

ICTIR'17, October 1–4, 2017, Amsterdam, The Netherlands.

978-1-4503-4490-6/17/10…$15.00
DOI: http://dx.doi.org/10.1145/3121050.3121052

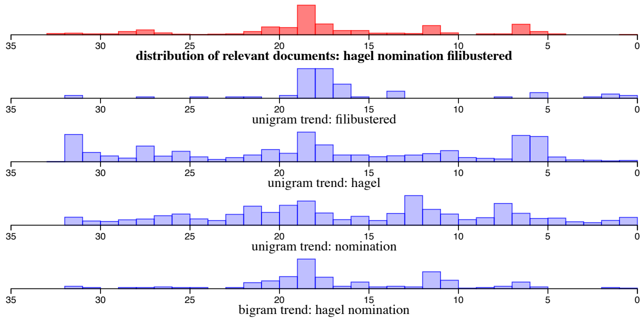


Figure 1: The temporal distribution of relevant documents (top row, in red) and unigram/bigram query trends (remaining rows, in blue) for MB127 ("hagel nomination filibustered") from the TREC 2013 Microblog Track. Informally, our problem can be characterized as using the blue distributions to predict the red distribution.

The main contribution of this work is the exploration of temporal collection statistics of query terms (what we call query trends) for temporal ranking. To our knowledge, our focus on such query term statistics is novel. Experimental evaluations on standard tweet test collections show that our proposed methods are significantly more effective than competitive baselines. Furthermore, detailed studies of different feature combinations show the extent to which different types of temporal signals impact retrieval effectiveness.

2 BACKGROUND AND RELATED WORK

We begin with an overview of related work on modeling temporal dynamics for document ranking and related tasks. Then we provide some technical details about recent work on temporal ranking to set up comparisons with our proposed methods.

2.1 Temporal Information Retrieval

There is a long thread of research exploring the role of temporal signals in search [5–9, 12, 17, 20], and it is well established that for certain tasks, better modeling of the temporal characteristics of queries and documents can lead to higher retrieval effectiveness.

For example, Jones and Diaz [10] studied the temporal profiles of queries, classifying queries as atemporal, temporally ambiguous, or temporally unambiguous. They showed that the temporal distribution of retrieved documents can provide an additional source of evidence to improve rankings. Building on this, Li and Croft [12] introduced recency priors that favor more-recent documents. Dakka et al. [5] proposed an approach to temporal modeling based on moving windows to integrate query-specific temporal evidence with lexical evidence. Efron et al. [7] presented several language modeling variants that incorporate query-specific temporal evidence. The most direct point of comparison to our work (as discussed in the introduction) is the use of non-parametric density estimation to infer the temporal distribution of relevant documents from an initial list of retrieved documents [8, 19]. Most recently, Rao et al. [17] proposed an end-to-end neural ranking model to integrate lexical and temporal signals, which has shown promising improvements over previous approaches.

There have been several other studies of time-based pseudo relevance feedback. Keikha et al. [11] represented queries and documents with their normalized term frequencies in the time dimension and used a time-based similarity metric to measure relevance. Craveiro et al. [4] exploited the temporal relationship between words for query expansion. Choi and Croft [3] presented a method to select time periods for expansion based on users' behaviors (i.e., retweets). Rao et al. [18] proposed a continuous hidden Markov model to identify temporal burst states in order to select better query expansion terms.

In addition to ranking, modeling temporal signals has also been shown to benefit related tasks such as behavior prediction [16], time-sensitive query auto-completion [21], and real-time event detection [1, 2]. For example, Radinsky et al. [16] built predictive models to learn query dynamics from historical user data.

One important difference between the above cited papers and our work lies in the source of the temporal signals. Temporal evidence in most previous studies comes either from behavior log data or from analyzing a candidate set of documents. We extend these approaches by incorporating the temporal distribution of collection term statistics as another source of temporal signal.

2.2 Temporal Modeling of Pseudo Trends

Consider the query-likelihood approach in the language modeling framework [15]: 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. Below, we discuss several ways to incorporate temporal signals within this general framework.

Recency Prior: One of the simplest way to let time influence ranking was proposed by Li and Croft [12], in the form of a document prior that favors recently published documents. If T_D is the timestamp associated with document D, $P(D)$ could take the form of an exponential distribution (with rate parameter $\lambda \geq 0$): $P(D) = \lambda e^{-\lambda T_D}$. Although previous studies have shown that recency priors increase overall effectiveness, they are by definition query-independent. This approach, however, is problematic because we know that dependencies between time and relevance vary from query to query [10].

Moving Window (WIN): Dakka et al. [5] proposed a query-specific way to combine lexical and temporal evidence in the language modeling framework by separating the two components: W_D, the document's content and T_D, the document's timestamp. This leads to the following derivation:

$$\begin{aligned} P(D|Q) &= P(W_D, T_D|Q) && (1) \\ &= P(T_D|W_D, Q)P(W_D|Q) && (2) \\ &\sim P(T_D|Q)P(W_D|Q) && (3) \end{aligned}$$

where the last step follows from Eq. (2) if we assume independence between content and temporal evidence. More generally, we take the view that there are two sources of evidence we need to integrate in document ranking: $P(R|W_D, Q)$, based on document content, and $P(R|T_D, Q)$, based on temporal evidence.

The content relevance term $P(R|W_D, Q)$ can be modeled through a standard query-likelihood model [15]. The temporal relevance term can be estimated through the temporal distribution of documents retrieved by query Q. Since this temporal distribution is estimated from the initial retrieved documents, we call this the *pseudo trend* approach, in contrast with our *query trend* methods. To estimate the pseudo trend, Dakka et al. [5] adopted a moving window technique to group retrieved document into discrete bins based on the publication time of the documents.

Kernel Density Estimation (KDE): Efron et al. [8] extended the pseudo trend approach by inferring a continuous density function using kernel density estimation. As discussed in the introduction, the theoretical motivation for modeling the distribution of initial retrieved documents is what Efron et al. [8] call the temporal cluster hypothesis: that relevant documents tend to cluster together in time, in the same way that van Rijsbergen's "classic" cluster hypothesis suggests that documents relevant to a query Q cluster in term space. An example of such cluster distributions can be found in the first row of Figure 1.

In the KDE approach, each document is modeled as a Gaussian kernel estimator and associated with a weight to denote its importance. Efron et al. proposed four weighting schemas: *uniform*, *score-based*, *rank-based*, and *oracle*. Uniform weights assume that each document contributes equally, score-based weights are derived from normalized retrieval scores, and rank-based weights are computed from an exponential decay function of the rank positions of the documents. Finally, oracle weights come from performing KDE directly on the relevant documents (i.e., from human judgments). Of course, we do not know the distribution of relevant documents at query time, but the oracle weights quantify the effectiveness upper bound of KDE-based techniques.

Once we obtain an estimate of the pseudo trend, we can then compute the temporal relevance term for each document given its publication timestamp. This feature is further integrated with the lexical relevance term in a simple log-linear model as follows:

$$\begin{aligned}\log P_\alpha(R|D,Q) = Z_\alpha &+ (1-\alpha)\log P(R|W_D,Q)\\ &+ \alpha \log P(R|T_D,Q)\end{aligned} \tag{4}$$

where Z_α is a normalization constant. These scores are then used to rerank documents from the initial query.

3 APPROACH

3.1 Temporal Modeling of Query Trends

Instead of attempting to estimate the distribution of relevant documents from the results of an initial query—what we've called pseudo trend approaches in the previous section—we adopt the alternative approach of directly leveraging the temporal distribution of query term statistics, which we call query trends. This approach has the obvious advantage of not requiring an initial retrieval; temporal term statistics can be gathered and efficiently compressed for low-latency lookup [20] as part of the indexing process.

Intuitively, we would expect to find more relevant documents in temporal intervals where the query terms are bursty. We illustrate this in Figure 1 for topic MB127 ("hagel nomination filibustered") from the TREC 2013 Microblog Track, as described in the introduction. The top row shows the actual distribution of relevant documents, which is the target of our prediction and of course not known at query time. The remaining rows show the query trends of the unigrams "filibustered", "hagel", "nomination", and the bigram "hagel nomination".[1] As we might expect, there are correspondences between peaks in the query trends and the actual distribution of relevant documents—for example, the few days when the unigram "filibustered" occurs most frequently are also when most of the relevant documents are clustered.

Of course, not all query trends are created equal. In the example in Figure 1, we see that the distribution of the unigram "nomination" is less predictive of the distribution of relevant documents. Overall, we find that less bursty terms are less useful, a notion we can formally capture by computing the entropy of the distribution. Given the counts of a particular unigram or bigram $t = \{c_1, c_2, ..., c_n\}$ across various time intervals (e.g., days), its entropy can be computed as follows:

$$\text{Entropy}(t) = -\sum_i \frac{c_i}{C}\log\frac{c_i}{C} \tag{5}$$

where $C = \sum_i c_i$. Lower entropy indicates a less uniform distribution and thus more bursty behavior.

From the query trends we can derive a family of features for a learning-to-rank model. There is, however, one additional complication we need to address: queries vary in length, which means that different queries have different numbers of unigram and bigram query trends. This is problematic since the linear feature-based model we use assumes a fixed number of features. We address this issue in a more principled manner in the next section, but here we introduce features based on the unigram and bigram with the lowest entropy (thus, the largest burstiness). We call these the *representative* unigram and bigram query trend, respectively.

From the basic concepts introduced above, we propose the following features:

- The relative entropy of the representative unigram. The relative entropy reflects the burstiness of a unigram query trend, computed as the absolute difference between the unigram entropy and the maximum entropy. The maximum entropy is computed by assuming a uniform distribution over term counts. Note that queries can have different timespans (because each is associated with a different query time), and thus the maximum entropy is query-dependent; computing relative entropy normalizes for the effects of different query timespans.
- The relative entropy of the representative bigram. This feature is computed in exactly the same manner as described above, except on bigram query trends.
- Estimated density at the document's timestamp from the query trend of the representative unigram. This feature is document-dependent. First, we perform kernel density estimation over the representative query unigram. Then, for the particular document that we are scoring, we compute the estimated density at the document's timestamp.
- Estimated density at the document's timestamp from the query trend of the representative bigram. This is similar to above, except with bigrams.

In Section 3.3, we detail how these features are integrated into the final ranking model.

[1] The other query bigram "nomination filibustered" is ignored in this analysis because it does not occur with sufficient frequency (based on a simple threshold).

	Description
N_q	number of queries
N_p	number of sample points per query
N	$N_q \cdot N_p$, number of sample points across all queries
N_u	max. number of unigrams per query (default 10)
N_b	max. number of bigrams per query (default 10)
Y_i	$N_p \times 1$, densities computed from relevant docs for query i
Y	$N \times 1$, concatenation of densities $(Y_1, ..., Y_i, ..., Y_{N_q})$
U	$N \times N_u$, densities computed from unigram trends
B	$N \times N_b$, densities computed from bigram trends
E_u	$N_q \times N_u$, normalized relative unigram entropies
E_b	$N_q \times N_b$, normalized relative bigram entropies
R	$N_q \times 1$, ratio of max unigram to bigram entropy
w_i^u	$N_u \times 1$, weight vector for unigrams of query i
w_i^b	$N_b \times 1$, weight vector for bigrams of query i

Table 1: Notation Table.

3.2 Regression on Query Trends

The above feature engineering approach tries to predict the distribution of relevant documents via a single representative unigram or bigram query trend. An alternative is to integrate evidence from *all* unigram and bigram query trends. Such an approach, however, can be a double-edged sword. On the one hand, we observe that for many topics, the distribution of relevant documents has many peaks. In these cases, it is unlikely that a single unigram or bigram query trend is sufficient to reconstruct the reference distribution. Such cases would seemingly benefit from integrating multiple sources of evidence to overcome the limited signal from any individual query trend. On the other hand, we see that some query trends have low or even negative correlations with the actual distribution of relevant documents (e.g., query terms that aren't important to the information need). In these cases, the query trends merely introduce noise into the prediction. How to balance these two factors is a question we explore.

The basic idea behind our regression-based method is to predict the actual query distribution by integrating all unigram and bigram query trends. When a query arrives, we can apply the entropy computations and kernel density estimations on all query terms. Suppose we have computed an entropy of e_t and a kernel density function of f_t for each term t. We can then attempt to fit the actual density of relevant documents Y (which is obtained by KDE on the distribution of relevant documents) as follows:

$$Y \approx \sum_t w_t f_t \tag{6}$$

where weight w_t is a function of entropy e_t and our goal is to learn this mapping function.

Note that approximating a continuous function from multiple kernel density functions is difficult, so instead we sample the distributions at fixed intervals. Now this model transforms into a non-linear regression problem. Given the unigram entropies E_u, bigram entropies E_b, unigram densities U at the sample points, and bigram densities B at the sample points, our task is to predict the densities Y at the same points. For more details about symbols used in this section, please refer to Table 1.

Two questions need to be answered in this non-linear regression problem. First, how to determine the importance of each term in contributing to the estimated density? Based on our observations, we find that terms with larger normalized entropies, i.e., a larger difference between its absolute entropy and the entropy of a uniform distribution, are more likely to reflect the true distribution of relevant documents. Therefore, we formulate the mapping from entropy to weights via an exponential increasing function, $w_t = \exp(\theta \cdot e_t) - 1$, where e_t is the normalized entropy of term t with its value ranging from zero to one. A term with zero normalized entropy would have zero weight, and thus can be ignored. The parameter θ controls the exponential rate. We use α for unigrams and β for bigrams as θ below.

The second question is how to differentiate contributions of unigrams from those of bigrams. For some queries, unigram query trends are more predictive, while for others, bigram trends are more predictive. How to evaluate their contributions for different queries is one key aspect of our model. To this end, for each query, we assign a weight $u_i \in [0, 1]$ to denote its unigram contribution; the corresponding bigram weight would be $1 - u_i$. We link the normalized unigram weight u_i to the entropy ratio R_i (which is the ratio of the maximum normalized unigram to bigram entropy for query i) by observing correlations between these two factors in training data. This mapping is normalized by a logistic function:

$$u_i = \text{logistic}(R_i, \gamma) = \frac{1}{1 + \exp(-\gamma R_i)} \tag{7}$$

where

$$R_i = \frac{\max_u E_i^u}{\max_b E_i^b} - 1 \tag{8}$$

and γ is a parameter to be estimated.

Intuitively, R_i greater than zero implies that the maximum normalized unigram entropy is larger than the maximum normalized bigram entropy. In this case, the logistic function would assign a unigram weight $u_i > 0.5$, and so unigrams would contribute more to the density estimate than bigrams. Finally, we desire that the integrated densities approximate the actual query density Y for each query i:

$$Y_i \approx u_i U_i w_i^u + (1 - u_i) B_i w_i^b \tag{9}$$

where w_i^u and w_i^b are weight vectors of unigrams and bigrams of query i, respectively. Overall, we sum up the square loss between ground truth densities Y_i and the estimated densities $\hat{Y}_i$ over all queries, plus some regularization terms. The final loss function L is formulated as follows:

$$\begin{aligned} L = \sum_{i=1}^{N_q} &||Y_i - (u_i U_i (e^{\alpha E_i^u} - 1)^T + (1 - u_i) B_i (e^{\beta E_i^b} - 1)^T)||^2 \\ &+ \lambda(\alpha^2 + \beta^2 + \gamma^2) \end{aligned} \tag{10}$$

where R_i and u_i are defined above.

Note that this model has three parameters (α, β, and γ) to be estimated, which are the weights of the entropy mapping function and the logistic function. Since the loss L is differentiable with respect to the three parameters, we can optimize the parameters using gradient-based methods. By constituting the logistic function into the overall loss function L, the gradients with respect to the parameters are computed as follows:

	Description
1	QL score
	Density estimate from:
2	KDE over initial retrieved docs (uniform)
3	KDE over initial retrieved docs (score-based)
4	KDE over initial retrieved docs (rank-based)
5	KDE over relevant docs (oracle)
Section 3.1	
6	Relative entropy of representative unigram
7	Relative entropy of representative bigram
	Density estimate from:
8	KDE of representative unigram distribution
9	KDE of representative bigram distribution
Section 3.2	
10	Density estimate from query trend regression model

Table 2: Summary of all features.

$$\text{term} = 2 \cdot \left(Y_i - (u_i U_i (e^{\alpha E_i^u} - 1)^T + (1 - u_i) B_i (e^{\beta E_i^b} - 1)^T)\right)$$

$$\frac{\partial L}{\partial \alpha} = -\sum_{i=1}^{N_q} \left(u_i \cdot \text{term}^T \cdot U_i \cdot (E_i^u \cdot e^{\alpha E_i^u})^T\right) + 2\lambda\alpha$$

$$\frac{\partial L}{\partial \beta} = -\sum_{i=1}^{N_q} \left((1 - u_i) \cdot \text{term}^T \cdot B_i \cdot (E_i^b \cdot e^{\beta E_i^b})^T\right) + 2\lambda\beta$$

$$\frac{\partial L}{\partial \gamma} = \sum_{i=1}^{N_q} \left(\text{term}^T \cdot \left(-U_i \cdot (e^{\alpha E_i^u} - 1)^T + B_i \cdot (e^{\beta E_i^b} - 1)^T\right) \cdot R_i \text{logistic}(R_i, \gamma) \cdot (1 - \text{logistic}(R_i, \gamma))\right) + 2\lambda\gamma$$

After solving the objective, we learn two mappings: an exponential mapping from entropy to term weight $w^t = \exp(\theta e) - 1$, and a logistic mapping from ratio to unigram weight $u = \text{logistic}(R, \gamma)$. We are then able to estimate densities for queries in the test data:

$$\hat{Y}_i = u_i U_i w_i^u + (1 - u_i) B_i w_i^b \qquad (11)$$

Finally, the estimated density $\hat{Y}_i$ serves as a feature in the final evidence combination approach (more details below).

3.3 Pulling Everything Together

To recap, we have introduced three families of features for modeling temporal evidence: KDE applied to initial retrieved documents [8] (Section 2.2), features derived from query trends (Section 3.1), and density estimates from a query trend regression model (Section 3.2). In total, we have ten features, including query-likelihood for capturing content relevance, which are summarized in Table 2.

As previously discussed, we integrate all these features in a linear feature-based ranking model [13]. The general form of such a model, extended from Eq. (4), is as follows:

$$S_d = \sum_i \alpha_i \cdot F_i(d, q) \quad \text{s.t.} \sum_i \alpha_i = 1. \qquad (12)$$

Naturally, we would like to understand the relative contributions of each type of feature, but it does not make sense to exhaustively explore all possible combinations. Thus, we took the middle road and explored a number of interesting feature set combinations, summarized in Table 3:

Method	Features
QL	1
IRD$_u$	1, 2
IRD$_s$	1, 3
IRD$_r$	1, 4
QT	1, 6–9
QT + IRD$_r$	1, 4, 6–9
Reg	1, 10
Reg + IRD$_r$	1, 4, 10
Oracle	1, 5

Table 3: Summary of different feature combinations.

- Different weighting schemes for KDE applied to the initial retrieved documents. These are the same experimental conditions in Efron et al. [8] and Rao et al. [19]. For convenience, these models are referred to as IRD$_u$ (uniform weights), IRD$_s$ (score-based weights), and IRD$_r$ (rank-based weights). Previous experiments [19] show that rank-based weights are the most effective overall, and thus for subsequent configurations we only use rank-based weights.
- Query trend features as a group (QT) and query trend features combined with KDE on the initial retrieved documents with rank-based weights (QT + IRD$_r$).
- Query trend regression (Reg) and query trend regression combined with KDE on the initial retrieved documents with rank-based weights (Reg + IRD$_r$).

Note that use of the IRD$_r$ features requires an initial retrieval, and thus we lose the efficiency advantage of feature combinations that use only query trends.

4 EVALUATION

4.1 Experimental Setup

We evaluated our proposed methods on Twitter test collections from the TREC 2013 and 2014 Microblog Tracks (60 topics and 55 topics, respectively). Both use the Tweets2013 collection, which consists of approximately 243 million tweets crawled from Twitter's public sample stream between February 1 and March 31, 2013. NIST assessors provided relevance judgments on a three-point scale ("not relevant", "relevant", "highly relevant") but in this work we treated both higher grades as "relevant". We removed all retweets in our experiments since they are by definition not relevant according to the assessment guidelines.

To rule out the effects of different preprocessing strategies during collection preparation (i.e., stemming, stopword removal, etc.), we used the open-source implementations of tweet search provided by the TREC Microblog API[2] to retrieve up to 1000 tweets per topic using query likelihood (QL) for scoring. On this candidate set of documents we applied our various methods for reranking. Following the TREC Microblog Tracks, we used average precision (AP) and precision at 30 (P30) to measure effectiveness.

[2] https://github.com/lintool/twitter-tools

ID	Method		Odd-Even		Even-Odd		Cross	
			AP	P30	AP	P30	AP	P30
1	Query Likelihood (QL) [15]		0.271	0.475	0.357	0.564	0.315	0.520
2	Recency prior [12]		0.277	0.499^{1}	0.359	0.574	0.313	0.534^{1,4}
3	Moving Window (WIN) [5]		0.283^{1}	0.487^{1}	0.358	0.567	0.319	0.527
4	KDE [8]	IRD$_u$	0.273	0.481	0.350	0.566	0.308	0.515
5		IRD$_s$	0.274	0.487^{1}	0.353	0.577^{1}	0.314	0.530^{1,4}
6		IRD$_r$	0.288^{1,4,5}	0.517^{1,3,4,5}	0.360	0.588^{1,2,3,4}	0.327^{1,2,4,5}	0.552^{1,2,3,4,5}
7	This work	**QT**	0.278	0.492^{1,4}	0.367^{1,4,5}	0.587^{1,2,3,4}	0.320	0.530^{1,4}
8		**Reg**	0.276	0.488^{1}	0.366^{1,4,5}	0.576^{1}	0.329^{1,2,4,5}	0.535^{1,4}
9		**QT-IRD$_r$**	0.290^{1,2,4,5}	0.522^{1,2,3,4,5}	**0.370**^{1,2,3,4,5}	**0.598**^{1,2,3,4,5}	0.328^{1,2,4,5}	0.565^{1,2,3,4,5}
10		**Reg-IRD$_r$**	**0.302**^{1,2,3,4,5,6}	**0.535**^{1,2,3,4,5}	0.368^{1,2,3,4,5}	0.596^{1,2,3,4,5}	**0.332**^{1,2,3,4,5}	**0.566**^{1,2,3,4,5}
11	Oracle		0.314^{1,2,3,4,5,6}	0.536^{1,2,3,4,5,6}	0.382^{1,2,3,4,5,6}	0.636^{1,2,3,4,5,6}	0.349^{1,2,3,4,5,6}	0.586^{1,2,3,4,5,6}

Table 4: Results from the TREC 2013/14 Microblog Track test collections: "Odd-Even" represents training on odd topics and testing on even topics; "Even-Odd" represents the opposite; "Cross" represents four-fold cross validation. Superscripts indicate the row indexes from which the metric differences are statistically significant ($p < 0.05$).

In our experiments, we examined four different ways of splitting the test collections into training and test sets:

- First, we trained on odd-numbered topics from the TREC 2013 and 2014 Microblog Tracks (57 topics) and evaluated on even-numbered topics (58 topics).
- Second, we swapped the training/test splits: training on even-numbered topics and testing on odd-numbered topics.
- Third, we performed four-fold cross validation across all topics.
- Finally, we performed a series of trials in which we randomly selected half the topics for training and used the remaining for testing. Results across multiple trials are aggregated.

We used coordinate ascent in RankLib[3] to learn the parameters in Eq. (12), optimizing and evaluating on the same metric.

Several baselines were used as points of comparison to our proposed methods. Query likelihood (QL) [15] was used as a lexical baseline. Temporal baselines included:

- Li and Croft's recency prior method [12].
- The moving window method of Dakka et al. [5].
- The kernel density estimation (KDE) methods of Efron et al. [8] with uniform weights (IRD$_u$), score-based weights (IRD$_s$), and rank-based weights (IRD$_r$).

In addition, we also include the KDE oracle as a reference upper bound. In this condition, we apply kernel density estimation over the distribution of the relevant documents based on human assessor judgments. This characterizes how much temporal signal can be extracted to improve relevance ranking, at least with this class of density estimation techniques.

To build the query trend features, we needed to precompute collection frequencies across time windows for the entire vocabulary. We aggregated term statistics and worked with query trends at the day granularity—that is, each term's trend is represented by an integer array of size 59, where each integer denotes the collection frequency for a single day. By discarding terms with a collection frequency lower than five, we extracted a total of 2.3 million unigrams and 23.1 million bigrams from the Tweets2013 collection. These term statistics are compressed with PForDelta [20], down to a size of 0.26 GB for unigrams and 2.2 GB for bigrams. The average decoding time of the compressed term statistics is 5.1 μs per unigram and 5.8 μs per bigram on a commodity server. Due to the efficient compression, we are able to load the term statistics into memory to estimate query trend features very quickly.

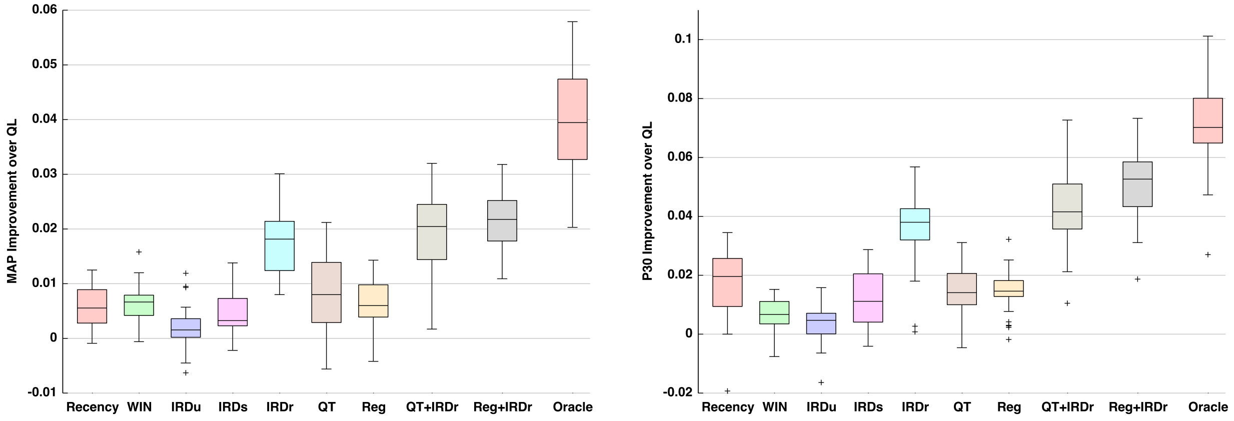
4.2 Effectiveness of Temporal Models

Results of our experiments are summarized in Table 4. Each row denotes an experimental condition (numbered for convenience): the third column "Odd-Even" represents training on odd-numbered topics and testing on even-numbered topics; "Even-Odd" represents the opposite; "Cross" represents four-fold cross validation. The best result for each setting is in bold. We compared each method against all lexical and temporal baselines for statistical significance using Fisher's two-sided, paired randomization test [22]. Superscripts indicate the row indexes from which the metric differences are statistically significant ($p < 0.05$).

First, we observe that most temporal baselines (Recency, WIN, IRD$_s$, and IRD$_r$) outperform the lexical baseline in terms of P30, but generally not in terms of AP, suggesting that they are better suited to improving early precision. Among the temporal baselines, IRD$_u$ performs consistently the worst and IRD$_r$ outperforms the rest. Note that while IRD$_s$ and IRD$_r$ both place more weight on top-ranked documents, the gap in effectiveness comes from the fact that the retrieved scores of the top-ranked documents are generally quite similar. Thus, score normalization does not introduce sufficient bias to help us distinguish the high-ranking documents.

Second, we see that our query trend methods (QT and Reg) significantly outperform the lexical baselines in most conditions, suggesting that signals captured from temporal collection statistics are beneficial to relevance ranking. While these "vanilla" query trend methods alone do not significantly improve over the temporal baselines, combining them with the pseudo trend methods (as in QT+IRD$_r$ and Reg+IRD$_r$) yields a boost in effectiveness. These ensemble methods are consistently more effective than the best-performing temporal baseline IRD$_r$. They also come close to the upper bound (oracle) in some conditions, especially for P30. For the Reg+IRD$_r$ model, features 1, 4, and 10 (query likelihood, IRD$_r$, Reg

[3] https://sourceforge.net/p/lemur/wiki/RankLib/



(a) AP on TREC 2013/14

(b) P30 on TREC 2013/14

Figure 2: Box-and-whiskers plots summarizing how much each temporal model outperforms the QL baseline across 30 random trials (half for training, half for testing) on the TREC 2013/14 Microblog Track test collections.

features) received weights 0.84, 0.10, and 0.06 in the Odd-Even split, respectively, which shows that the different sources of temporal evidence are complementary.

In the above experiments, we noticed variance in effectiveness under different conditions, depending on how the test collections are split into training/test sets. Our random split experiments were designed to factor out noise from this issue. In each trial, we trained on half of the topics (randomly selected) and evaluated on the other half. We then computed the effectiveness differences between each technique and the QL baseline. These differences, collected over 30 trials, are summarized in box-and-whiskers plots in Figure 2 for all temporal approaches. We show the distribution of effectiveness differences in terms of AP (left) and P30 (right). 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 each method, better than metrics from any single arbitrary split.

From Figure 2, it is clear that IRD$_r$ outperforms all baselines as well as the raw query trend approaches (QT and Reg). The ensemble approaches (QT+IRD$_r$ and Reg+IRD$_r$) yield further improvement over IRD$_r$, with Reg+IRD$_r$ coming out higher. Although we did not observe a statistically significant difference between our best ensemble method (Reg+IRD$_r$) and the best baseline (IRD$_r$) in our previous experiments, the box plots show that the effectiveness gains of Reg+IRD$_r$ are more consistent, This is especially true for P30 (right side of Figure 2): the median of Reg+IRD$_r$ is above 0.05 whereas IRD$_r$ has a median below 0.04. Another observation is that the bottom of the Reg+IRD$_r$ box is still above the top of the IRD$_r$ box, meaning that the top 75% of Reg+IRD$_r$ runs were better than the bottom 75% of IRD$_r$ runs. Although it is difficult to definitively conclude statistical significance from these experiments, quantifying the variance associated with arbitrary training/test splits provides additional evidence supporting the effectiveness of our proposed methods.

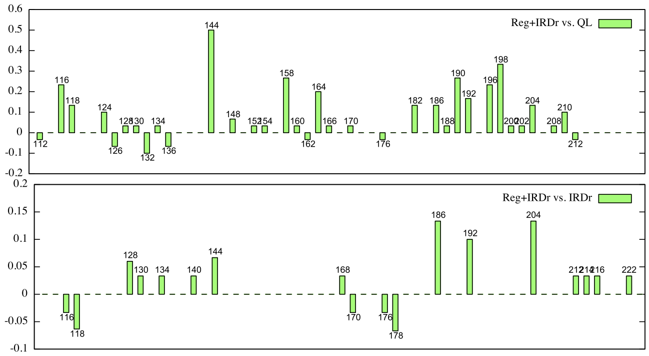


Figure 3: Per-topic improvements of the ensemble model Reg+IRD$_r$ compared to the QL baseline and IRD$_r$ method.

4.3 Per-topic Analysis

In order to gain a better understanding of how different temporal features contribute to effectiveness in temporal ranking, we performed a topic-by-topic analysis along with an in-depth examination of the various component distributions. Due to a lack of space, here we present only results comparing the best-performing ensemble model (Reg+IRD$_r$) against the lexical baseline QL and the temporal baseline IRD$_r$. In Figure 3, we show per-topic differences as a bar chart, measured in terms of P30 on the even topics from the TREC 2013/14 Microblog Track test collections.

From the top bar chart in Figure 3, we can see that the ensemble model (Reg+IRD$_r$) improves over the QL baseline for most of the topics; there are only a few topics where effectiveness decreases (and not by much). From the bottom bar chart, we see that Reg+IRD$_r$ improves over IRD$_r$ alone, which confirms that the regression model contributes additional signal over kernel density estimation alone.

In Figure 4 we take a closer look at the best-performing topic, MB144 “downtown abbey actor turnover”. The top row shows the distribution of relevant documents (in red), the second row shows

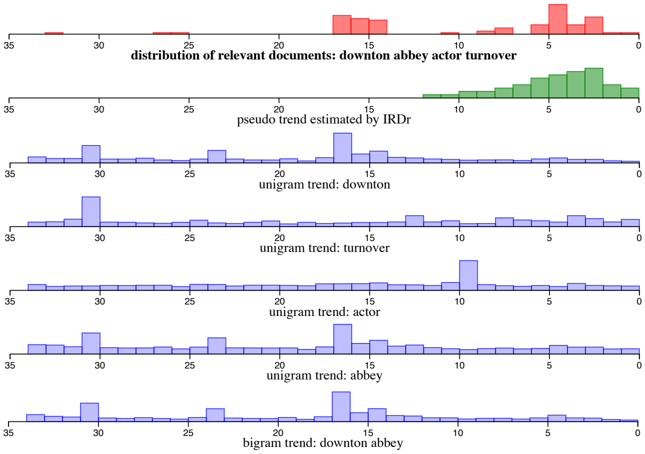


Figure 4: Analysis of MB144 ("downtown abbey actor turnover") from the TREC 2013 Microblog Track. Rows show: distribution of relevant documents (red), pseudo trend based on KDE (green), and query trends (blue).

the distribution inferred from the pseudo trend using IRD$_r$ (in green), and the remaining rows show the query trends (in blue). Clearly, we can see that the distribution of relevant documents has two peaks, one around days 3–5 and the other around days 14–16. We can observe that the pseudo trend from IRD$_r$ is able to capture the burst of relevant documents at days 3–5. We also see a strong correlation between the query trends (unigrams "downtown", "abbey", and the bigram "downtown abbey") and the ground truth relevance distribution at days 14–16. Thus, the combination of pseudo trend and query trend features allows us to nicely recover this multimodal distribution, which is affirmed by the large improvements for this topic compared to both QL and IRD$_r$. In addition, the regression model is able to smooth out noise from non-important terms "actor" and "turnover". This observation is confirmed in many other topics, like MB192 "whooping cough epidemic" and MB204 "sotomayor, prosecutor, racial comment", where we also observe strong correlations between query trends and the ground truth relevance distributions.

We also examined topics where effectiveness decreased with respect to IRD$_r$, such as MB116 "Chinese computer attacks" and MB118 "Israel and Turkey reconcile". We found that the representative query trends (the unigram "Chinese" for MB116 and the bigram "and Turkey" for MB118) are very different from the distribution of relevant documents, and thus our methods infer an inaccurate distribution. No approach is perfect, but overall our per-topic analysis affirms the effectiveness of our query trend methods.

5 CONCLUSION

Quite obviously, the temporal distribution of relevant documents provides an important signal for temporal ranking. As an alternative to previous pseudo trend methods that analyze the results of an initial query to infer this distribution, we propose query trend methods that attempt to make predictions directly from the temporal collection statistics of query terms. Experiments show that these sources of evidence are complementary, and the regression method appears to be more effective than the feature-based approach. Although query trend methods alone, which do not require an initial retrieval, improve over a lexical baseline, combining query trends with pseudo trends yields the best results. This ensemble approach, however, *does* require an initial retrieval, which negates the performance advantages of query trend methods. Costly approaches that involve actually searching the collection appear to provide temporal signals that we currently cannot obtain from the temporal collection statistics of query terms alone.

6 ACKNOWLEDGMENTS

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, with additional contributions from the U.S. National Science Foundation under CNS-1405688. Any findings, conclusions, or recommendations expressed do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Chao Zhang, Liyuan Liu, Dongming Lei, Quan Yuan, Honglei Zhuang, Tim Hanratty, and Jiawei Han. 2017. TrioVecEvent: Embedding-Based Online Local Event Detection in Geo-Tagged Tweet Streams. In *KDD*. 595–604.

[2] Chao Zhang, Guangyu Zhou, Quan Yuan, Honglei Zhuang, Yu Zheng, Lance Kaplan, Shaowen Wang, and Jiawei Han. 2016. GeoBurst: Real-Time Local Event Detection in Geo-Tagged Tweet Streams. In *SIGIR*. 513–522.

[3] Jaeho Choi and W. Bruce Croft. 2012. Temporal Models for Microblogs. In *CIKM*. 2491–2494.

[4] Olga Craveiro, Joaquim Macedo, and Henrique Madeira. 2014. Query Expansion with Temporal Segmented Texts. In *ECIR*. 612–617.

[5] Wisam Dakka, Luis Gravano, and Panagiotis G. Ipeirotis. 2012. Answering General Time-Sensitive Queries. *TKDE* 24, 2 (2012), 220–235.

[6] Anlei Dong, Ruiqiang Zhang, Pranam Kolari, Jing Bai, Fernando Diaz, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. 2010. Time is of the Essence: Improving Recency Ranking Using Twitter Data. In *WWW*. 331–340.

[7] Miles Efron and Gene Golovchinsky. 2011. Estimation Methods for Ranking Recent Information. In *SIGIR*. 495–504.

[8] Miles Efron, Jimmy Lin, Jiyin He, and Arjen de Vries. 2014. Temporal Feedback for Tweet Search with Non-Parametric Density Estimation. In *SIGIR*. 33–42.

[9] Jonathan L. Elsas and Susan T. Dumais. 2010. Leveraging Temporal Dynamics of Document Content in Relevance Ranking. In *WSDM*. 1–10.

[10] Rosie Jones and Fernando Diaz. 2007. Temporal Profiles of Queries. *TOIS* 25, 3 (2007), Article 14.

[11] Mostafa Keikha, Shima Gerani, and Fabio Crestani. 2011. TEMPER: A Temporal Relevance Feedback Method. In *ECIR*. 436–447.

[12] Xiaoyan Li and W. Bruce Croft. 2003. Time-Based Language Models. In *CIKM*. 469–475.

[13] Donald Metzler and W. Bruce Croft. 2007. Linear Feature-Based Models for Information Retrieval. *Information Retrieval* 10, 3 (2007), 257–274.

[14] Gilad Mishne, Jeff Dalton, Zhenghua Li, Aneesh Sharma, and Jimmy Lin. 2012. Fast Data in the Era of Big Data: Twitter's Real-Time Related Query Suggestion Architecture. In *SIGMOD*. 1147–1157.

[15] Jay M. Ponte and W. Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. In *SIGIR*. 275–281.

[16] Kira Radinsky, Krysta Svore, Susan Dumais, Jaime Teevan, Alex Bocharov, and Eric Horvitz. 2012. Modeling and Predicting Behavioral Dynamics on the Web. In *WWW*. 599–608.

[17] Jinfeng Rao, Hua He, Haotian Zhang, Ferhan Ture, Royal Sequiera, Salman Mohammed, and Jimmy Lin. 2017. Integrating Lexical and Temporal Signals in Neural Ranking Models for Social Media Search. In *SIGIR Workshop on Neural Information Retrieval (Neu-IR)*.

[18] Jinfeng Rao and Jimmy Lin. 2016. Temporal Query Expansion Using a Continuous Hidden Markov Model. In *ICITR*. 295–298.

[19] Jinfeng Rao, Jimmy Lin, and Miles Efron. 2015. Reproducible Experiments on Lexical and Temporal Feedback for Tweet Search. In *ECIR*. 755–767.

[20] Jinfeng Rao, Xing Niu, and Jimmy Lin. 2016. Compressing and Decoding Term Statistics Time Series. In *ECIR*. 675–681.

[21] Milad Shokouhi and Kira Radinsky. 2012. Time-Sensitive Query Auto-Completion. In *SIGIR*. 601–610.

[22] Mark D. Smucker, James Allan, and Ben Carterette. 2007. A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. In *CIKM*. 623–632.