# Pseudo Test Collections for Training and Tuning Microblog Rankers

Richard Berendsen
r.w.berendsen@uva.nl

Manos Tsagkias
e.tsagkias@uva.nl

Wouter Weerkamp
w.weerkamp@uva.nl

Maarten de Rijke
derijke@uva.nl

ISLA, University of Amsterdam, Amsterdam, The Netherlands

## ABSTRACT

Recent years have witnessed a persistent interest in generating pseudo test collections, both for training and evaluation purposes. We describe a method for generating queries and relevance judgments for microblog search in an unsupervised way. Our starting point is this intuition: tweets with a hashtag are relevant to the topic covered by the hashtag and hence to a suitable query derived from the hashtag. Our baseline method selects all commonly used hashtags, and all associated tweets as relevance judgments; we then generate a query from these tweets. Next, we generate a timestamp for each query, allowing us to use temporal information in the training process. We then enrich the generation process with knowledge derived from an editorial test collection for microblog search.

We use our pseudo test collections in two ways. First, we tune parameters of a variety of well known retrieval methods on them. Correlations with parameter sweeps on an editorial test collection are high on average, with a large variance over retrieval algorithms. Second, we use the pseudo test collections as training sets in a learning to rank scenario. Performance close to training on an editorial test collection is achieved in many cases. Our results demonstrate the utility of tuning and training microblog search algorithms on automatically generated training material.

## Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval Models

## Keywords

Simulation, pseudo test collections, learning to rank, microblog retrieval

## 1. INTRODUCTION

Modern information retrieval (IR) systems have evolved from single model based systems to intelligent systems that learn to combine uncertain evidence from multiple individual models [10, 22]. The effectiveness and flexibility of such systems has led to wide adoption in IR research. A key contributor to the success of such systems is the learning phase, i.e., the training set they are given for learning. Training sets have to be tailored to the task at hand and, in contrast to the systems themselves, do not generalize to other tasks. This characteristic requires compiling task-specific training sets, which is a time consuming and resource intensive process, as it usually involves human labor. Automating the process of compiling training sets has obvious advantages in reducing costs, while it simultaneously increases the size of the training set. This observation has led to a persistent interest in finding ways for generating so-called *pseudo test collections*, which consist of a set of queries, and for each query a set of relevant documents (given some document set). In this paper, we consider the problem of generating pseudo test collections for microblog search.

Microblog search is the task of finding information in microblogs, such as Facebook status updates, Twitter posts, etc. The task became popular with the advent of social media and is distinct from web search and from blog search due mainly to its real-time nature, the very limited length of microblog posts and the use of "microblog language," e.g., hashtags, mentions, which can provide useful information for retrieval purposes. In 2011 the Text REtrieval Conference (TREC) launched the Microblog track aimed at developing a test collection from Twitter data and evaluating systems' performance on retrieving—given a query and time-stamp—relevant and interesting tweets in a simulated real-time scenario. Several participants approached the task using learning to rank methods for combining evidence from multiple rankers [21]. This approach to microblog search comes natural because of the many dimensions available for ranking microblog posts, e.g., recency, user authority, content, existence of hyperlinks, hashtags, retweets. For training a learning to rank-based system at the TREC 2011 Microblog track, participants used a traditional supervised method: many manually labeled data for compiling a training set. What if we could generate the required training sets automatically? In 2012 a second edition of the Microblog track was organized. This gives us the opportunity to compare what yields better learning to rank performance: training on the 2011 relevance assessments, or training on automatically generated ground truth?

Our starting point is the following intuition, based upon the observation that hashtags tend to represent a topic in the Twitter domain: *From tweets $T_h$ associated with a hashtag $h$, select a subset of tweets $R_h \subseteq T_h$ that are relevant to an unknown query $q_h$ related to $h$.* We build on this intuition for creating a training set for microblog rankers. To this end, we take several steps, each raising research questions. First, we select hashtags $h$ and associated relevant tweets $R_h$. Can we just select all hashtags and use all their associated tweets? In microblog search, time is important: what is considered relevant to a query may change rapidly over time. A


*SIGIR'13,* July 28–August 1, 2013, Dublin, Ireland.

microblog query, then, has a timestamp, and relevant tweets must occur prior to this timestamp. As for a query, the topic a *hashtag* is associated with may change over time. Can we exploit this analogy, and label hashtags with a timestamp, regarding tweets prior to this timestamp as relevant? Another well-known aspect of microblog posts is that they often contain casual conversation that is unlikely to be relevant to a query. Can we improve generated training sets by selecting interesting tweets and hashtags associated with such tweets? Once we have selected a hashtag $h$ and a set of tweets $R_h$, how do we generate a query $q_h$ related to $h$?

The main contribution of this paper is a set of methods for creating pseudo test collections for microblog search. These collections are shown to be useful as training material for tuning well-known retrieval methods from the literature, and for optimizing a learning to rank method. In particular, we contribute: (1) unsupervised pseudo test collection generation methods; (2) a supervised pseudo test collection generation method, where we learn what are interesting tweets from TREC Microblog track assessments; (3) insights into the sensitivity of our methods to parameter settings.

## 2. PROBLEM DEFINITION

Below, we consider a number of instantiations of our pseudo test collection generator. For the purposes of the TREC Microblog track, a *test collection for microblog search* consists of queries with timestamps and a set of relevant documents for these queries. A *pseudo test collection for microblog search* consists of a set of queries $\mathcal{Q}$, in which each query $q \in \mathcal{Q}$ is associated with a timestamp $q_t$ and a set of relevant documents $R_q$. Given this definition, there are three main steps for generating a pseudo test collection for microblog search: generating (a) the query; (b) the query timestamp; and (c) a set of relevant tweets for the query.

We start from the following intuition: *From the tweets $T_h$ that contain a hashtag $h$, we can select tweets $R_h$ that are relevant to an unknown query $q_h$ related to $h$.* In the next section, we present three methods to generate a pseudo test collection. Each method selects hashtags and for every hashtag $h$ it selects tweets $R_h$ from $T_h$ that will act as relevant tweets to a suitable query related to $h$. In §4, we present a technique for generating queries from $R_h$.

## 3. SELECTING HASHTAGS AND TWEETS

We propose four solutions to selecting hashtags and tweets for inclusion in a pseudo test collection. **(A) Random:** A sanity check baseline against our hypothesis that hashtags are good sources for generating pseudo test collections. Collections are created by randomly sampling a set of relevant tweets for each topic, without replacement. All these random collections are of a fixed size, equal to our largest hashtag-based pseudo test collection. **(B) Hashtags:** A naive method that serves as baseline in our experiments and that considers all hashtags and tweets to be equally important (§3.1). **(C) Hashtags-T:** A method that creates a test collection in the microblog retrieval sense, in which queries have timestamps (§3.2). **(D) Hashtags-TI:** A method that aims at capturing interestingness in tweets. Interesting tweets should contain good candidate terms for a query. We present a method with which we can estimate from example queries and relevant tweets the probability of interestingness of a tweet (§3.3).

### 3.1 Hashtags: All hashtags, tweets are equal

We select all hashtags, with a single requirement: that they are mentioned in a reasonable amount of tweets, $m$ (Algorithm 1). There are three reasons for this lower bound: (i) it reflects a certain consensus about the meaning of a hashtag; (ii) we generate our queries based on word distributions in these tweets: for this to work reliably, we need a reasonable amount of tweets, see §4; (iii) we train a learning to rank retrieval algorithm on our pseudo test collection; we hypothesize that it would benefit from a relative large set of positive training examples, see §5. We normalize hashtags by lowercasing them and removing any non-alphanumeric characters. In all our experiments, we set $m = 50$. The pseudo test collection generated by Algorithm 1 is called *Hashtags*.

**Algorithm 1:** Generating collection *Hashtags*

$H \longleftarrow \{h : |T_h| >= m\}$;
**for** $h \in H$ **do**
  $R_h \longleftarrow T_h$;
  Generate query $q_h$ from $R_h$; `// See §4`
**end**

### 3.2 Hashtags-T: Generating timestamps

Microblog search is sensitive to the query issue time because of the real time nature of tweets. To generate a timestamp for a query related to a hashtag $h$, we make an analogy between search volume over time for a query and publishing volume over time for tweets with $h$. Our assumption is that users often issue queries for trending topics because they want to monitor developments, similar to certain types of blog search [28]. We generate a timestamp for hashtag $h$ just after peaks in publishing volume. In this way, our generated queries will be about trending topics. In addition, we keep a large amount of tweets from $T_h$, while discarding a limited number, after $h$ stops trending. In collections that span a considerable period of time, re-occurring topics, such as Christmas or Super Bowl, may quite likely be observed. In this case, one may want to assign multiple issue times for a query, depending on the number of observed peaks. Our corpus (see §5) covers a relatively short period, and we assign only one issue time to every query sampled.

In detail, our query issue time generation works as follows. First, we group the time span of the collection in 8-hours bins. Then, for each hashtag, we count how many relevant documents belong to each bin; this results in generating the hashtag's timeseries. In our setting, timeseries are short and sparse; our peak detection method aims at coping with this challenge. We find the bin with the most counts and resolve ties by taking the earliest date. This approach allows us to return a peak even for very sparse timeseries. We call the pseudo test collection generated by Algorithm 2: *Hashtags-T*.

### 3.3 Hashtags-TI: Selecting interesting tweets

Consider the following tweet: "Hey follow me here #teamfollowback #justinbieber." We hypothesize that this tweet would not be useful for sampling terms for topics labeled #teamfollowback or #justinbieber or as a relevant document for these topics. To avoid selecting such tweets, we rank tweets by their probability of interestingness and keep the best $X$ percent. We think of a tweet

**Algorithm 2:** Generating collection *Hashtags-T*

$H' \longleftarrow \{h : |T_h| \geq m\}$;
**for** $h \in H'$ **do**
  Generate timestamp $t(h)$; `// See §3.2`
  $R_h \longleftarrow \{\tau : \tau \in T_h \text{ and } t(\tau) \leq t(h)\}$;
**end**
$H \longleftarrow \{h : h \in H' \text{and } |R_h| \geq m\}$;
**for** $h \in H$ **do**
  Generate query $q_h$ from $R_h$; `// See §4`
**end**

as interesting if it carries some information and could be relevant to a query. We use a set of criteria to capture interestingness and present a method to learn from example queries and relevant documents from an editorial collection.

Let $C_1, C_2, \ldots, C_n$ be random variables associated with the criteria and let $I_\tau := I(\tau) = 1$ be the event that a tweet is interesting. We estimate $P(I_\tau|C_1 = c_1, \ldots, C_n = c_n)$, or, shorthand: $P(I_\tau|c_1, \ldots, c_n)$. Following Bayes' rule, we have

$$P(I_\tau|c_1, \ldots, c_n) = \frac{P(c_1, \ldots, c_n|I_\tau)P(I_\tau)}{P(c_1, \ldots, c_n)}, \quad (1)$$

where $P(I_\tau)$ is the a-priori probability that a tweet is interesting, $P(c_1, \ldots, c_n|I_\tau)$ is the likelihood of observing the evidence given that a tweet is interesting, and $P(c_1, \ldots, c_n)$ is the probability of observing the evidence. The crucial step is to estimate $P(c_1, \ldots, c_n|I_\tau)$. We hypothesize that tweets that are known to be relevant to a query are interesting and estimate $P(c_1, \ldots, c_n|I_\tau)$ with $P(c_1, \ldots, c_n|R_\tau)$, where $R_\tau$ is the event that a tweet is relevant to a query in an editorial collection. We use the TREC Microblog 2011 qrels for this estimation. Since we do not have enough relevant tweets to estimate the full joint probability, we assume conditional independence of $c_i$ given that a tweet is relevant:

$$P(I_\tau|c_1, \ldots, c_n) \approx \frac{\left(\prod_i P(c_i|R_\tau)\right) P(I_\tau)}{P(c_1, \ldots c_n)}. \quad (2)$$

Since we rank tweets by interestingness we do not have to estimate $P(I_\tau)$. Instead, we have:

$$\text{rank}_\tau(P(I_\tau|c_1, \ldots, c_n)) = \text{rank}_\tau\left(\frac{\sum_i \log(P(c_i|R_\tau))}{P(c_1, \ldots, c_n)}\right). \quad (3)$$

Most of the criteria we use have discrete distributions. For those that do not, we bin their values in $B$ bins; we set $B = 10$. To avoid rejecting a tweet on the basis of one measurement that did not occur in any of the relevant tweets, we add one observation to every bin of every $P(c_i|R_\tau)$ distribution. For $P(c_1, \ldots, c_n)$ we use the empirical distribution of all tweets in the collection. After selecting the best X percent of tweets, we again filter out hashtags that have less than 50 interesting tweets. We build this method on top of our pseudo test collection *Hashtags-T*, only ranking the tweets in this collection and keeping the best 50% of them. We call the pseudo test collection generated by Algorithm 3 *Hashtags-TI*.

**Algorithm 3:** Generating collection *Hashtags-TI*

```
H'' ⟵ {h : |T_h| ≥ m};
for h ∈ H'' do
    Generate timestamp t(h);                    // See §3.2
    T_{h,t} ⟵ {τ : τ ∈ T_h and t(τ) ≤ t(h)};
end
H' ⟵ {h : h ∈ H'' and |T_{h,t}| ≥ m};
T ⟵ ∪_{h∈H'} T_{h,t};
// Rank tweets by probability of being
   interesting
Rank T by P(I_τ|c_1, ..., c_n(τ));          // See eq. 3
Let T_I be the top X percent of this ranking;
for h ∈ H' do
    R_h ⟵ T_{h,t} ∩ T_I;
end
H ⟵ {h : h ∈ H' and |R_h| ≥ m};
for h ∈ H do
    Generate query q from R_h;                  // See §4
end
```

The criteria we use build on textual features (*density* and *capitalization*) and microblog features (*links*, *mentions*, *recency*). Each criterion is discussed below. The marginal distributions $P(c_i|R_\tau)$ of three criteria are shown in Fig. 1 as white histograms. They overlap with black histograms of all tweets in our *Hashtags* pseudo test collection. These criteria have different distributions over relevant tweets and over tweets that have a hashtag, which motivates our idea to keep tweets with high probability of interestingness.

**Links.** The existence of a hyperlink is a good indicator of the content value of a tweet. TREC Microblog 2011 defines interestingness of a tweet as whether a tweet contains a link [21]. Also, a large fraction of tweets are pointers to online news [19]. Tweets with links are likely to include terms that describe the linked web page, rendering them good surrogates for query terms [5].

**Mentions.** Tweets with mentions (@username) signify discussions about the hashtag's topic. This type of tweet is likely to be noisy because of their personal character. They may, however, bring in query terms used by a niche of people.

**Tweet length.** Document length has been shown to matter in retrieval scenarios [35]. Short tweets are less likely to contain terms useful for query simulation, see Fig. 1 (Center) for the distribution of tweet length.

**Density.** A direct measure for probing a tweet's content quality is the density score [20]. Density is defined as the sum of tf-idf values of non-stopwords, divided by the number of stopwords they are apart, squared:

$$Density(\tau) = \frac{K}{K-1}\sum_{k=1}^{K-1}\frac{weight(w_k) + weight(w_{k+1})}{distance(w_k, w_{k+1})^2},$$

where $K$ is the total number of non-stopwords terms in tweet $\tau$, $w_k$ and $w_{k+1}$ are two adjacent keywords in $\tau$. $weight(\cdot)$ denotes the term's tf-idf score, and $distance(w_k, w_{k+1})$ denotes the distance between $w_k$ and $w_{k+1}$ in number of stopwords. Fig. 1 (Left) shows the distribution of density scores of tweets.

**Capitalization.** The textual quality of tweets can partially be captured through the use of capitalization [41]. Words in all capitals are considered shouting and an indication of low quality. The ratio of capitals may indicate the quality of the text. Fig. 1 (Right) shows the distribution of the fraction of capital letters over tweet length.

**Direct.** A tweet is *direct* if it is meant to be a "private" message to another user (i.e., the tweets starts with *@user*).

## 4. GENERATING QUERIES

Sampling query terms is a challenging step in the process of automatically generating pseudo test collections. Azzopardi et al. [3] propose several methods for sampling query terms from web documents for known-item search, while Asadi et al. [2] avoid the problem by using anchor texts. Neither approach is applicable in the microblog setting due to a lack of both redundancy in the tweets and anchor texts. Terms in tweets usually occur at most once, but if not, this is often a signal for spam [26]. Probabilistic sampling methods that boost terms occurring with high probability are less likely to return good candidates for query terms. Tf-idf methods emphasize rare terms, which are likely to be spelling mistakes or descriptive of online chatter (e.g., "looooool") in our setting.

The log-likelihood ratio (LLR) is suitable for our problem of sampling terms from (tweets associated with) a given hashtag [25]. LLR is defined as the symmetric Kullback-Leibler divergence of the expected and observed term probabilities in two corpora. Terms are ranked by how discriminative they are for both corpora. For

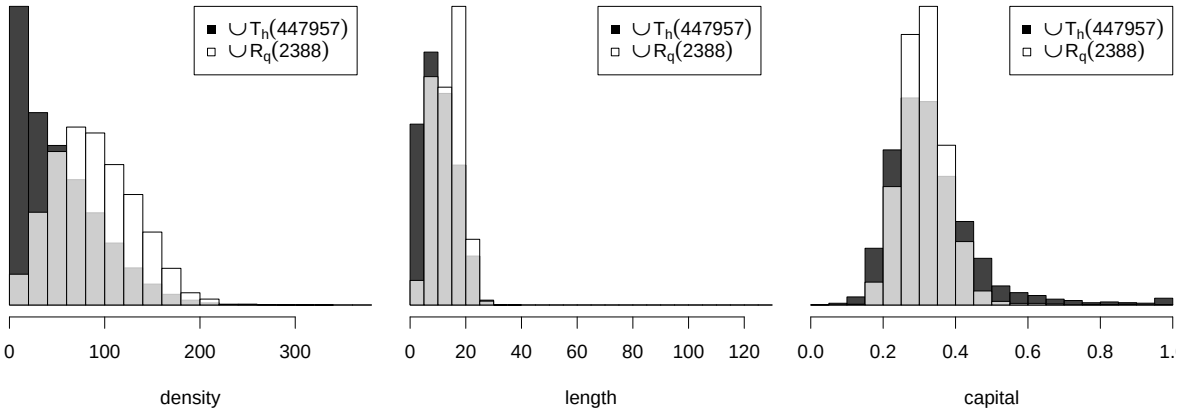


**Figure 1: Distribution of (Left) density scores, (Center) tweet length, and (Right) capitalization. Where the histograms for the tweets associated with hashtags (dark grey) and for the TREC MB 2011 relevant tweets (white) overlap, the color is light grey.**

our purposes, we set one corpus to be a set of tweets associated with hashtag $h$ and the other to consist of the rest of the tweets in the collection. Stopwords, spam terms, or terms indicative of online chatter will rank lower because they occur in both corpora with roughly the same frequency.

Let $T$ be a collection of tweets $\tau$, $R_h$ a set of relevant tweets associated with a hashtag $h$ resulting from one of the sampling methods described in §3, and $w$ a term in $\tau$. For every $w \in \bigcup_{\tau \in R_h} \tau$ we compute LLR($w$), given corpora $R_h$ and $B = T \setminus R_h$:

$$LLR(w) = 2 \cdot \left[ O_{R_h}(w) \log \frac{O_{R_h}(w)}{E_{R_h}(w)} + O_B(w) \log \frac{O_B(w)}{E_B(w)} \right],$$

where $O_{R_h}(w) = tf(w, R_h)$ and $O_B(w) = tf(w, B)$ are the observed term frequencies of $w$ in $R_h$ and $B$, respectively. $E_{R_h}(w)$ and $E_B(w)$ are the expected values of the term frequency of $w$ in $R_h$ and $B$, respectively.

For every hashtag $h$, terms are ranked in descending order of their log-likelihood ratio score. We remove terms if one of the following pertains: (1) The term is equal to the hashtag up to the '#' character. In this case, all tweets in $T_h$ contain the hashtag, making the query very easy for all rankers, which then leads to a learning to rank method having a hard time distinguishing between rankers. (2) The term occurs in fewer than 10 documents. We generate queries that consist of the top-$K$ ranked terms. For all pseudo test collections described in §3 we set $K = 10$. For our most promising method, we examine the impact of this parameter by generating queries of length 1, 2, 3, 5, and 20.

# 5. EXPERIMENTAL SETUP

The main research question we aim to answer is: *What is the utility of our test collection generation methods for tuning retrieval approaches and training learning to rank methods?*

**Parameter tuning.** We do parameter sweeps for some retrieval runs on different (pseudo) test collections, see Table 1 for details. We ask: (a) What is better in terms of retrieval performance: tuning on a different editorial test collection or tuning on a pseudo test collection? We answer by calculating how far performance obtained by tuning on either collection is from optimal performance for each retrieval model. (b) Do scores between a pseudo test collection and an editorial test collection correlate better than scores between editorial test collections? We answer by calculating Kendall's tau and expected loss in effectiveness.

**Learning to rank.** We compare the utility of test collections as training material for a learning to rank algorithm. We ask: (a) What is better in terms of retrieval performance: training on a different editorial test collection, or training on a pseudo test collection? We answer by calculating the difference in retrieval performance between training on each. (b) Which of our pseudo test collections is most useful? (c) Do learning to rank algorithms trained on pseudo test collections outperform the best individual feature?

**Parameter sensitivity.** We also analyze parameter sensitivity of our methods, focusing on two parameters. First, in generating *Hashtag-TI* (§3), we keep the best $X = 50\%$ percent of tweets. How sensitive are our results to this method to different values for X? We try these values: 20, 40, 60, and 80. In all our experiments,

**Algorithm 4:** Training an LTR system on a pseudo test collection, and testing it on an editorial collection

```
for i = 1 → N do
    // -- Training phase: --
    Generate the pseudo test collection;
    for each ranker do
        Sweep parameters on the pseudo test collection;
        Randomly sample parameter vector to use
        from winners;
    end
    for q ∈ Q do
        Merge the ranked lists into M_q;
        Let positive training examples ← R_{q,h} ∩ M_q;
        Randomly sample |R_{q,h}| negative training examples
        from M_q \ R_{q,h};
    end
    Learn a LTR model on the training set;
    // -- Testing phase: --
    for each ranker do
        Run on test topics using the sampled parameter
        vector;
    end
    Run the learned LTR model on the test set;
end
```

**Table 1:** Retrieval algorithms tuned.

| | Ranker | Description | Parameter | Values | cf. Literature |
|---|---|---|---|---|---|
| Indri 5.1 | LM | Language modeling with Dirichlet smoothing | $\mu$ | $\{50, 150, \ldots, 10050\}$ | [42] |
| | Tf-idf | Indri's implementation of tf-idf | k1 | $\{0.2, 0.4, \ldots, 3\}$ | [33] |
| | | | b | $\{0, 0.05 \ldots, 1\}$ | |
| | Okapi | Okapi, also known as BM25 [35] | k1 | $\{0, 0.2, \ldots, 3\}$ | [33] |
| | | | b | $\{0, 0.05 \ldots, 1\}$ | |
| | | | k3 | $\{0\}$ | |
| | BOW | Boolean ordered window | Window size | $\{1, 2 \ldots, 15, \text{inf}\}$ | |
| | BUW | Boolean unordered window | Window size | $\{1, 2 \ldots, 15, \text{inf}\}$ | |
| Terrier 3.5 | Tf-idf | Terrier's implementation of tf-idf | b | $\{0, 0.05 \ldots, 1\}$ | [33] |
| | PL2 | A Divergence from Randomness (DFR) model [1] | c | $\{0.5, 1, 5, 10\}$ | [9] |
| | DFR-FD | Terrier's DFRee model with a document score modifier that takes into account co-occurence within a window | Window size | $\{2, 3 \ldots, 15\}$ | [31] |
| | QE | Terrier's implementation of query expansion | No. documents | $\{1, 5, 10, 20, 30, 50\}$ | [23] |
| | | | No. terms | $\{1, 5, 10, 20, 30, 50\}$ | |

when we generate queries, we keep the top 10 terms of the ranking produced by LLR (§4). For our best pseudo test collection, we ask how parameter tuning results and learning to rank performance is influenced by different query length. We try query lengths 1, 2, 3, 5, and 20.

## 5.1 Dataset and preprocessing

We use the publicly available dataset from the TREC 2011 and 2012 Microblog tracks. It covers two weeks of Twitter data, from January 24, 2011–February 8, 2011, consisting of approximately 16 million tweets. We perform a series of preprocessing steps on the content of tweets. We discard non-English tweets using a language identification method for microblogs [6]. Exact duplicates are removed; among a set of duplicates the oldest tweet is kept. Retweets are discarded; in ambiguous cases, e.g., where comments were added to a retweet, we keep the tweet. Punctuation and stop words are removed using a collection-based stop word list, but we keep hashtags without the '#' character. After preprocessing we are left with 4,459,840 tweets, roughly 27% of all tweets. Due to our aggressive preprocessing, we miss 19% of the relevant tweets per topic, on average. Our 10 retrieval models avoid using future evidence by using per topic indexes. For completeness, we note that our stopword list and the idf-weights in the density feature were computed on the entire collection. Pseudo test collections and both TREC microblog test collections also contain tweets from the entire collection. For generating queries, we index the collection with Lucene without stemming or stopword removal.

Table 2 lists statistics of the pseudo test collections generated with the methods described in §3, as well as statistics of the collections generated by choosing different values for X. The Hashtags-TI-QL{1,2,3,5,20} pseudo test collections are of the same proportions as *Hashtags-TI*, apart from the query length. We have listed only one of fifteen Random collections, but these are all of the same proportions: about as large as the *Hashtags* collection.

**Table 2:** Statistics for pseudo test collection generated from our methods and the TREC Microblog 2011 track.

| Collection | Topics | | | | Relevant documents | | | |
|---|---|---|---|---|---|---|---|---|
| | # | Max | Min | Avg. length | # | Max | Min | Avg. |
| TREC MB 2011 | 49 | 6 | 1 | 3.4 | 2,965 | 178 | 1 | 60.5 |
| TREC MB 2012 | 59 | 7 | 1 | 2.9 | 6,286 | 572 | 1 | 106.5 |
| Random-1 | 1,888 | 10 | 10 | 10 | 462560 | 245 | 245 | 245.0 |
| Hashtags | 1,888 | 10 | 10 | 10 | 462,013 | 16,105 | 50 | 244.7 |
| Hashtags-T | 891 | 10 | 10 | 10 | 212,377 | 9,164 | 50 | 238.4 |
| Hashtags-TI | 481 | 10 | 10 | 10 | 98,586 | 4,949 | 50 | 205.0 |
| H-TI-X20 | 175 | 10 | 10 | 10 | 32,221 | 3661 | 50 | 184.1 |
| H-TI-X40 | 392 | 10 | 10 | 10 | 75,287 | 4804 | 50 | 192.1 |
| H-TI-X60 | 576 | 10 | 10 | 10 | 121,639 | 5277 | 50 | 211.2 |
| H-TI-X80 | 740 | 10 | 10 | 10 | 166,489 | 6956 | 50 | 225.0 |

## 5.2 Learning to rank

We follow a two-step approach to learning to rank, outlined in Algorithm 4. First, we run several retrieval algorithms, then we rerank all retrieved tweets. For retrieval, we use the retrieval algorithms listed in Table 1, optimized for MAP [24, 34] after tuning on a training collection. In case of ties among parameter vectors for a ranker, we randomly sample a parameter vector. We also use a parameter free retrieval algorithm, DFRee [1]. For re-ranking, we compute three groups of features.

**Query-tweet features.** These are features that have different values for each query-tweet pair. We subdivide these as follows. *Rankers*: the raw output of each retrieval algorithm. For LM, BOW and BUW we transform the raw output $X$ by taking the exponent: $\exp(X)$. *Ranker meta features*: the number of rankers that retrieved the tweet, the maximal, average, and median reciprocal rank of the tweet over all rankers. *Recency*: query-tweet time difference decay, computed as $\exp(t(\tau) - q_t)$, where $t(\tau)$ is the timestamp of the tweet and $q_t$ the timestamp of the query. We linearly normalize query-tweet features over all retrieved tweets for the query.

**Query features.** These are features which have the same value for every retrieved tweet within the same query. We use *Query clarity*, a method for probing the semantic distance between the query and the collection [11]. We linearly normalize query features over the set of retrieved tweets for all queries.

**Tweet features.** These are features that have the same value for each tweet independent of the query. We use the *Quality criteria* listed in §3: *link*, *mentions*, *tweet length*, *density*, *capitalization*, and *direct*. We linearly normalize tweet features over the set of retrieved tweets for all queries.

To build a training set, one needs positive and negative training examples. Let $q \in Q$ be a query from the training collection, $R_q$ the set of relevant tweets for query $q$, and $M_q$ the set of all retrieved tweets for $q$. Then, for each query in the training collection we use $R_q \cap M_q$ as positive examples. To have a balanced training set, we randomly sample $|R_q|$ tweets as negative training examples from $M_q \setminus R_q$.

Next, we feed the training set to four state of the art learners:

(a) Pegasos SVM[1] [36, 37], (b) Coordinate ascent [27], (c) RankSVM [17], and (d) RT-Rank [29]. We set Pegasos SVM to optimize the area under the ROC curve using indexed sampling of training examples, with regularization parameter $\lambda = 0.1$ for a maximum of 100,000 iterations. We set coordinate ascent to optimize for MAP with $\epsilon = 0.0001$ and maximum step size of 3. We use line search to optimize each feature with uniform initialization and consider only positive feature weights without projecting points on the manifold. For RankSVM we set the cost parameter to 1 per query. In preliminary experiments, RT-Rank performed poorly and therefore we choose to leave it out from our report.

Recall that training sets are compiled using tuned rankers and that in case of ties between different parameter vectors for a ranker, a random vector is selected. When compiling test sets for TREC MB 2011 and TREC MB 2012 to evaluate the utility of a training set, we use the exact same parameter vectors, so that the same set of features are used for training and testing.

Algorithm 4 has randomness in several stages: (i) when generating the pseudo test collection (only in the case of the Random collections), (ii) when sampling a winning parameter setting for each feature, (iii) when randomly sampling negative training examples, and (iv) during model learning. To obtain a reliable estimate of of the performance when training on a pseudo test collection, this procedure is repeated $N = 10$ times, each time generating a new pseudo test collection (in the case of the Random test collection), selecting random parameter vectors, selecting random negative training examples, and training an LTR model.

## 5.3 Evaluation

We report on precision at 30 (P30) on binary relevance judgments. We choose P30 because it has been one of the main metrics in both the TREC 2011 and 2012 Microblog track. We also report on MAP, as it is a well understood and commonly used evaluation metric in information retrieval, allowing us to better understand the behavior of our pseudo test collections. Note that in the 2011 task, tweets had to be ordered by their publication date instead of by their relevance. Many top performing systems treated the task as normal relevance ranking and cut off their ranked lists at rank 30 [21]. In the 2012 track organizers decided to focus on ranking by relevance again, which is what we will focus on.

**Testing for statistical significance.** For each training collection, we run Algorithm 4 $N = 10$ times, giving rise to $N$ scores for each topic, for each collection. We report average performance and sample standard deviation over these iterations. To also gain insight if any differences between a pair of training collections would be observed on different microblog topics from the same hypothetical population of topics, we proceed as follows. We pick for each collection the iteration of Algorithm 4 which had the smallest *training* error on that collection. Then, we do a paired t-test over differences per topic as usual and report the obtained p-values. Statistically significant differences are marked as $^{\blacktriangle}$(or $^{\blacktriangledown}$) for significant differences for $\alpha = .001$, or $^{\triangle}$(and $^{\triangledown}$) for $\alpha = .05$.

# 6. RESULTS AND ANALYSIS

First, we report on our parameter tuning results; then on our learning to rank results. We also analyze parameter sensitivity with regard to the percentage of interesting tweets kept and query length.

## 6.1 Parameter tuning results

The main outcomes in this section will be correlations, to answer the question whether relative performance of parameter values on pseudo test collections correlates with relative performance of the same values on an editorial collection. We begin with two case studies to gain a better understanding of the behavior of our pseudo text collections. We sweep (a) the document length normalization parameter $b$ for Terrier's tf-idf implementation (Fig. 2(a)), and (b) $\mu$ for Indri's implementation of language modeling with Dirichlet smoothing (Fig. 2(b)). We only include one of our ten random pseudo test collections; all random collections behave similarly, for all retrieval systems and metrics.

Fig. 2(a) shows that on the TREC MB 2011 collection there is a general trend to prefer lower values of $b$, possibly because of the very small average document length, which, in turn, renders the deviation from the average length close to one. All pseudo test collections capture this trend, including the Random-1 pseudo test collection. The curves of the pseudo test collections are smoother than the curve obtained when tuning on the TREC MB 2011 topics; this is because the pseudo test collections have far more test topics. Pseudo test collections show differences in absolute scores, but most importantly, we are interested in whether pseudo test collection predictions that one parameter vector is better than another correlate to such predictions of editorial collections. Kendall's tau expresses exactly that correlation, see Table 3. In addition, we want to know the following: if we sample a random parameter vector from those predicted to yield optimal performance on a pseudo test collection, what will be the expected loss with regard to optimal performance on an editorial collection? Table 5 provides these quantities. Correlations are high across the board, and expected loss is low. All pseudo test collections can be used to reliably tune the $b$ parameter of Terrier's TF-IDF, even the Random-1 collection.

Turning to a second case study, Indri's language modeling algorithm with Dirichlet smoothing, we see a different picture (Fig. 2(b)). The TREC MB 2011 topics show a slightly decreasing trend for larger $\mu$ values. We believe this is due to the short document length; tweets after processing are few terms long, and therefore even small $\mu$ values overshadow the document term probability with the background probability. This trend is not entirely captured by the pseudo test collections. All have a short increase for low values of $\mu$ which is much less pronounced in the TREC MB 2011 curve. After that, all except the Random-1 collection show a decline, if only in the third digit. Correlations are fair (Table 4), but the Random-1 collection fails here. Expected loss is low across the board (Table 6).

Looking at the big picture, we average the correlations and expected loss figures in Tables 7 and 8 over all nine retrieval models from Table 1. For the hashtag based collections, correlations are high, with a large variance over systems. Expected loss is low. This indicates that pseudo test collections can be used to reliably and profitably tune parameters for a variety of well established retrieval algorithms, with more or less success depending on which model is being tuned. Tuning retrieval models on all hashtag based pseudo test collections is about as reliable as tuning on editorial test collections. For most retrieval algorithms, a Random collection cannot be recommended for tuning. Thus, the idea of grouping tweets by hashtag has value for creating pseudo test collections for tuning retrieval algorithms.

## 6.2 Learning to rank results

In this section we evaluate the usefulness of our pseudo test collections (PTCs) by training several learning to rank algorithms on them. In Tables 9 and 10, we report P30 and MAP performance on the TREC 2011 Microblog track topics. We compare training on our PTCs with training on the TREC 2012 Microblog track topics and indicate significant differences. In Tables 11 and 12, we report P30 and MAP performance on the TREC 2012 Microblog

[1] http://code.google.com/p/sofia-ml/

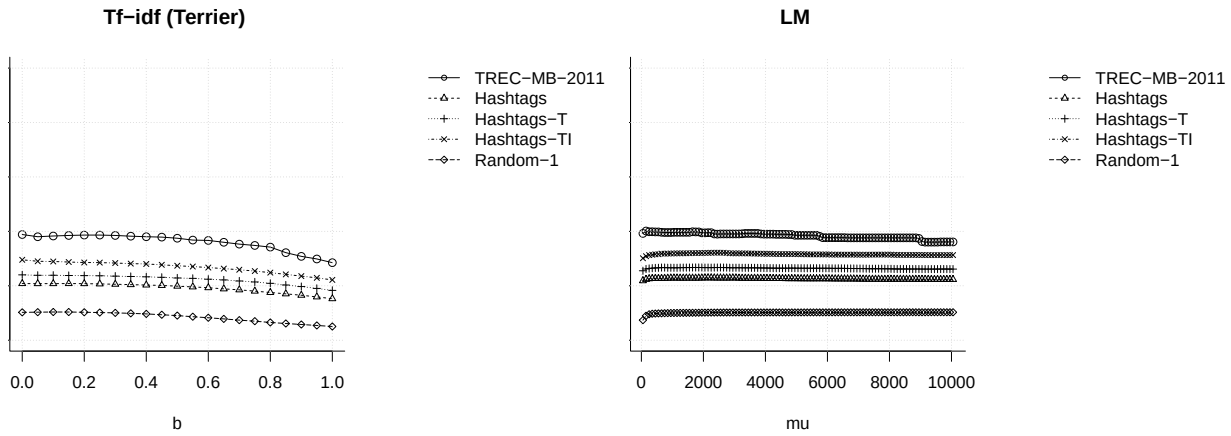


**Figure 2:** Sweeping the $b$ parameter of Tf-idf (Terrier) (2(a)) and the $\mu$ parameter of LM (2(b)). The x-axes have parameter values, the y-axes average P30 over the topics of the respective tuning collection.

**Table 3:** For Tf-idf (Indri), and P30, Kendall's tau correlations of parameter sweeps on several pseudo test collections with sweeps on TREC MB 2011 and 2012 collections. Kendall's tau between sweeps over TREC MB 2011 and 2012 is 0.85.

| Tune on | TREC MB 2011 | TREC MB 2012 |
|---|---|---|
| Random-1 | 0.18 | 0.20 |
| Hashtags | 0.87 | 0.86 |
| Hashtags-T | 0.90 | 0.86 |
| Hashtags-TI | 0.90 | 0.86 |

**Table 4:** For language modeling (LM), and P30, Kendall's tau correlations of parameter sweeps on several pseudo test collections with sweeps on TREC MB 2011 and 2012 collections. Kendall's tau between sweeps over TREC MB 2011 and 2012 is 0.61.

| Tune on | TREC MB 2011 | TREC MB 2012 |
|---|---|---|
| Random-1 | -0.87 | -0.65 |
| Hashtags | 0.65 | 0.78 |
| Hashtags-T | 0.69 | 0.77 |
| Hashtags-TI | 0.58 | 0.79 |

track topics, and compare training on our PTCs with training on the TREC 2011 Microblog track topics.

A first brief glance at all four tables tells us that training on an editorial test collection is in most cases (but not all) the best strategy. Still, if training on a PTC is not substantially and significantly worse, we may conclude that in the abscence of training data, pseudo test collections are a viable alternative.

A second glance at all four tables shows us that training on the Random PTC is always significantly outperformed by training on editorial collections. Still, in most cases, there is a hashtag based PTC on which training yields performance on par with training on editorial collections. This shows that there is added value in our idea of using hashtags to group tweets by topic.

Another phenomenon we can observe in all tables is that Pegasos has remarkably stable performance over pseudo test collections, compared to the other two learning to rank algorithms. With the exception of Hashtags-TI-QL1 and Random, it seems to be able to exploit the structure in any PTC to learn a function that yields performance comparable to a function learned on editorial training data. RankSVM, on the other hand, has unstable performance. Especially in Table 12 it refuses to work on anything but manually obtained ground truth. Coordinate Ascent, a remarkably simple learning to rank algorithm holds the middle ground. When queries are too short, as in Hashtags-TI-QL1, Hashtags-QL2 and Hashtags-TI-QL3 performance detoriates. When subsamples of tweets become too small (Hashtags-TI-X20) the same happens.

**Table 5:** For Tf-idf (Indri), expected loss in P30 performance of parameter sweeps on several training collections compared to optimal performance on TREC MB 2011 and 2012 collections.

| Tune on | TREC MB 2011 | TREC MB 2012 |
|---|---|---|
| TREC MB 2011 | - | 0.003 |
| TREC MB 2012 | 0.006±0.005 (3) | - |
| Random-1 | 0.029 | 0.024 |
| Hashtags | 0.012±0.002 (2) | 0.002±0.002 (2) |
| Hashtags-T | 0.002 | 0.000 |
| Hashtags-TI | 0.002 | 0.000 |

**Table 6:** For language modeling (LM), expected loss in P30 performance of parameter sweeps on several training collections compared to optimal performance on TREC MB 2011 and 2012 collections.

| Tune on | TREC MB 2011 | TREC MB 2012 |
|---|---|---|
| TREC MB 2011 | - | 0.010 |
| TREC MB 2012 | 0.006 | - |
| Random-1 | 0.039±0.001 (11) | 0.014±0.001 (11) |
| Hashtags | 0.010 | 0.001 |
| Hashtags-T | 0.013±0.004 (2) | 0.001±0.001 (2) |
| Hashtags-TI | 0.014 | 0.002 |

So which PTC is the best? All PTCs are significantly outperformed by training on an editorial collection in at least one of the conditions. Hashtags-TI-X60 and Hashtags-TI-X80 both yield best results in one case. Also, like Hashtags, Hashtags-T and Hashtags-TI are significantly outperformed in only a small number of cases.

Learning to rank only makes sense if improvements over individual retrieval algorithms can be obtained. Tables 13 and 14 show performance of the retrieval models we use as features. In the great majority of cases our hashtag based PTCs outperform the best feature. The Random collection never achieves this.

# 7. RELATED WORK

We describe two types of related work: (i) searching microblog posts and (ii) pseudo test collections.

**Searching microblog posts.** Microblog search is a growing research area. The dominant microblogging platform that most research focuses on is Twitter. Microblogs have characteristics that introduce new problems and challenges for retrieval [12, 40]. Massoudi et al. [26] report on an early study of retrieval in microblogs, and introduce a retrieval and query expansion method to account for microblog search challenges. Efron and Golovchinsky [13] inves-

**Table 7:** For all systems, and P30, Kendall's tau correlations of parameter sweeps on several pseudo test collections with sweeps on TREC MB 2011 and 2012 collections. Kendall's tau between sweeps over TREC MB 2011 and 2012 is 0.80.

| Tune on | TREC MB 2011 | TREC MB 2012 |
|---|---|---|
| Random-1 | 0.27±0.62 (2 NA) | 0.32±0.56 (1 NA) |
| Hashtags | 0.78±0.20 (1 NA) | 0.70±0.35 (1 NA) |
| Hashtags-T | 0.80±0.20 (1 NA) | 0.78±0.30 (1 NA) |
| Hashtags-TI | 0.75±0.26 (1 NA) | 0.81±0.23 (1 NA) |

**Table 8:** Over all retrieval models, average expected loss in P30 performance of parameter sweeps on several training collections compared to optimal performance on TREC MB 2011 and 2012 collections.

| Tune on | TREC MB 2011 | TREC MB 2012 |
|---|---|---|
| TREC MB 2011 | - | 0.003±0.006 |
| TREC MB 2012 | 0.003±0.003 | - |
| Random-1 | 0.021±0.021 | 0.013±0.012 |
| Hashtags | 0.005±0.006 | 0.004±0.006 |
| Hashtags-T | 0.004±0.005 | 0.002±0.005 |
| Hashtags-TI | 0.004±0.005 | 0.002±0.005 |

tigate the temporal aspects of documents on query expansion using pseudo relevance feedback. Naveed et al. [30] develop a retrieval model that takes into account document length and interestingness defined over a range of features.

In 2011, TREC launched the microblog search track, where systems are asked to return relevant and interesting tweets given a query [21]. The temporal aspect of Twitter and its characteristics, e.g., hashtags and existence of hyperlinks, were exploited by many participants, for query expansion, filtering, or learning to rank [21]. Whereas teams depended on self-constructed training data to train learning to rank systems during TREC 2011, most of these systems were successfully trained on the 2011 queries during TREC 2012. Teevan et al. [40] find that over 20% of Twitter queries contain a hashtag, an indication that hashtags may be good topical surrogates, which can be leveraged for building pseudo test collections.

**Pseudo test collections.** The issue of creating and using pseudo test collections is a longstanding and recurring theme in IR, see, e.g., [38, 39]. Several attempts have been made to either simulate human queries or generate relevance judgments without the need of human assessors for a range of tasks. Azzopardi et al. [3] simulate queries for known-item search and investigate term weighting methods for query generation. Their main concern is not to develop training material, but to determine whether their pseudo test collection generation methods ultimately give rise to similar rankings of retrieval systems as manually created test collections. Kim and Croft [18] generate a pseudo test collection for desktop search. Huurnink et al. [15] use click-through data to simulate relevance assessments. Later, they evaluate the performance of query simulation methods in terms of system rankings [16] and they find that incorporating document structure in the query generation process results in more realistic query simulators. Hofmann et al. [14] try to smooth noise from click-through data from an audio-visual archive's transaction log using purchase information of videos. We extend previous work on pseudo test collection generation [4] with a principled method to obtain good quality training material, using knowledge derived from editorial judgements.

Asadi et al. [2] describe a method for generating pseudo test collections for training learning to rank methods for web retrieval. Their methods build on the idea that anchor text in web documents is a good source for sampling queries, and the documents that these anchors link to are regarded as relevant documents for the anchor text (query). Our work shares analogies, but in the microblog setting, there is no anchor text to sample queries. Moreover, the temporal aspect of relevance plays a bigger role in microblog search.

Carterette et al. [7] investigate what the minimal judging effort is that must be done to have confidence in the outcome of an evaluation. Rajput et al. [32] present a method for extending the recall base in a manually created test collection. Carterette et al. [8] find that test collections with thousands of queries with fewer relevant documents considerably reduce the assessor effort with no appreciable increase in evaluation errors. This finding inspired us to come up with pseudo test collection generators that are able to produce large numbers of queries: while the signal produced by an individual query may be noisy, the volume will produce a signal that is useful for learning and parameter tuning.

**Table 9:** P30 performance on TREC MB 2011 topics for various LTR algorithms, trained on different collections. Features are tuned on MAP. Best run in bold. Indicated statistically significant differences are with regard to training on TREC-MB-2012. The best individual tuned ranker, DFR-FD, achieved 0.416.

| Train on | Pegasos | RankSVM | CA |
|---|---|---|---|
| TREC-MB-2012 | **0.446±0.001** | **0.436±0.004** | **0.447±0.004** |
| Random | 0.401±0.004$^{\triangledown}$ | 0.356±0.011$^{\blacktriangledown}$ | 0.378±0.006$^{\blacktriangledown}$ |
| Hashtags | 0.437±0.002 | 0.430±0.004 | 0.434±0.002 |
| Hashtags-T | 0.438±0.001 | 0.426±0.002 | 0.434±0.003 |
| Hashtags-TI | 0.437±0.001 | 0.406±0.007$^{\triangledown}$ | 0.429±0.002$^{\triangledown}$ |
| Hashtags-TI-X20 | 0.432±0.001 | 0.265±0.016$^{\blacktriangledown}$ | 0.383±0.016$^{\blacktriangledown}$ |
| Hashtags-TI-X40 | 0.435±0.001 | 0.361±0.006$^{\blacktriangledown}$ | 0.427±0.001$^{\triangledown}$ |
| Hashtags-TI-X60 | 0.438±0.001 | 0.415±0.004$^{\triangledown}$ | 0.431±0.003$^{\triangledown}$ |
| Hashtags-TI-X80 | 0.438±0.001 | 0.427±0.003 | 0.435±0.002 |
| Hashtags-TI-QL1 | 0.189±0.135$^{\blacktriangledown}$ | 0.034±0.015$^{\blacktriangledown}$ | 0.293±0.107 $^{\blacktriangledown}$ |
| Hashtags-TI-QL2 | 0.435±0.002 | 0.244±0.024$^{\blacktriangledown}$ | 0.421±0.047 $^{\blacktriangledown}$ |
| Hashtags-TI-QL3 | 0.443±0.001 | 0.240±0.012$^{\blacktriangledown}$ | 0.427±0.023 $^{\blacktriangledown}$ |
| Hashtags-TI-QL5 | 0.443±0.001 | 0.259±0.008$^{\blacktriangledown}$ | 0.428±0.005 $^{\triangledown}$ |
| Hashtags-TI-QL20 | 0.438±0.001 | 0.320±0.007$^{\blacktriangledown}$ | 0.432±0.004 |

**Table 10:** MAP performance on TREC MB 2011 topics for various LTR algorithms, trained on different collections. Features are tuned on MAP. Best performance per LTR algorithm in bold. Indicated statistically significant differences are with regard to training on TREC-MB-2012. The best individual tuned ranker, Tf-idf (Indri), achieved 0.357.

| Train on | Pegasos | RankSVM | CA |
|---|---|---|---|
| TREC-MB-2012 | **0.388±0.001** | 0.362±0.003 | **0.387±0.003** |
| Random | 0.348±0.004$^{\blacktriangledown}$ | 0.294±0.009$^{\blacktriangledown}$ | 0.319±0.009$^{\blacktriangledown}$ |
| Hashtags | 0.374±0.001 | 0.360±0.005 | 0.385±0.001 |
| Hashtags-T | 0.379±0.000 | 0.362±0.005 | 0.386±0.000 |
| Hashtags-TI | 0.383±0.001 | 0.344±0.003 | 0.377±0.002 |
| Hashtags-TI-X20 | 0.376±0.001$^{\triangledown}$ | 0.198±0.015$^{\blacktriangledown}$ | 0.332±0.013$^{\blacktriangledown}$ |
| Hashtags-TI-X40 | 0.380±0.001 | 0.300±0.007$^{\triangledown}$ | 0.373±0.001$^{\triangledown}$ |
| Hashtags-TI-X60 | 0.383±0.001 | 0.352±0.005 | 0.381±0.003 |
| Hashtags-TI-X80 | 0.381±0.000 | **0.363±0.002** | 0.384±0.002 |
| Hashtags-TI-QL1 | 0.141±0.125$^{\blacktriangledown}$ | 0.020±0.009$^{\blacktriangledown}$ | 0.202±0.119$^{\blacktriangledown}$ |
| Hashtags-TI-QL2 | 0.370±0.001$^{\triangledown}$ | 0.185±0.019$^{\blacktriangledown}$ | 0.360±0.046$^{\blacktriangledown}$ |
| Hashtags-TI-QL3 | 0.379±0.000 | 0.166±0.011$^{\blacktriangledown}$ | 0.366±0.026$^{\blacktriangledown}$ |
| Hashtags-TI-QL5 | 0.383±0.001 | 0.165±0.007$^{\blacktriangledown}$ | 0.372±0.006$^{\triangledown}$ |
| Hashtags-TI-QL20 | 0.383±0.001 | 0.231±0.008$^{\blacktriangledown}$ | 0.381±0.002 |

## 8. DISCUSSION AND CONCLUSION

Following the results of our experiments we list three main observations: (1) The Random pseudo test collection performs significantly worse than editorial collections in the retrieval experiments and shows low correlation to these collections in the tuning phase. (2) The top pseudo test collections are not significantly worse than editorial collections and show high correlation when tuning parameters. (3) Differences between various pseudo test collections on retrieval effectiveness are small.

**Table 11:** P30 performance on TREC MB 2012 topics for various LTR algorithms, trained on different collections. Features are tuned on MAP. Best performance per LTR algorithm in bold. Indicated statistically significant differences are with regard to training on TREC-MB-2011. The best individual tuned ranker, DFR-FD, achieved 0.351.

| Train on | Pegasos | RankSVM | CA |
|---|---|---|---|
| TREC-MB-2011 | **0.392±0.001** | **0.391±0.007** | 0.392±0.007 |
| Random | 0.341±0.003$^{\blacktriangledown}$ | 0.289±0.008$^{\blacktriangledown}$ | 0.314±0.006$^{\blacktriangledown}$ |
| Hashtags | 0.379±0.001 | 0.330±0.011$^{\triangledown}$ | 0.378±0.001$^{\triangledown}$ |
| Hashtags-T | 0.372±0.001$^{\triangledown}$ | 0.336±0.005$^{\triangledown}$ | 0.382±0.001 |
| Hashtags-TI | 0.381±0.001 | 0.326±0.007$^{\triangledown}$ | 0.393±0.002 |
| Hashtags-TI-X20 | 0.377±0.001 | 0.248±0.013$^{\blacktriangledown}$ | 0.353±0.008 |
| Hashtags-TI-X40 | 0.379±0.001$^{\triangledown}$ | 0.294±0.009$^{\blacktriangledown}$ | 0.389±0.001 |
| Hashtags-TI-X60 | 0.379±0.001 | 0.348±0.003$^{\triangledown}$ | **0.394±0.006** |
| Hashtags-TI-X80 | 0.373±0.001$^{\triangledown}$ | 0.337±0.006$^{\triangledown}$ | 0.388±0.007 |
| Hashtags-TI-QL1 | 0.198±0.089$^{\blacktriangledown}$ | 0.047±0.016$^{\blacktriangledown}$ | 0.291±0.059$^{\blacktriangledown}$ |
| Hashtags-TI-QL2 | 0.374±0.002$^{\triangledown}$ | 0.231±0.016$^{\blacktriangledown}$ | 0.377±0.035$^{\blacktriangledown}$ |
| Hashtags-TI-QL3 | 0.381±0.001 | 0.218±0.007$^{\blacktriangledown}$ | 0.382±0.016$^{\blacktriangledown}$ |
| Hashtags-TI-QL5 | 0.385±0.001 | 0.248±0.005$^{\blacktriangledown}$ | 0.391±0.004 |
| Hashtags-TI-QL20 | 0.380±0.001 | 0.302±0.005$^{\blacktriangledown}$ | 0.388±0.002 |

**Table 12:** MAP performance on TREC MB 2012 topics for various LTR algorithms, trained on different collections. Features are tuned on MAP. Best performance per LTR algorithm in bold. Indicated statistically significant differences are with regard to training on TREC-MB-2011. The best individual tuned ranker, DFR-FD, achieved 0.220.

| Train on | Pegasos | RankSVM | CA |
|---|---|---|---|
| TREC-MB-2011 | **0.245±0.000** | **0.245±0.004** | **0.246±0.004** |
| Random | 0.207±0.001$^{\blacktriangledown}$ | 0.159±0.005$^{\blacktriangledown}$ | 0.176±0.006$^{\blacktriangledown}$ |
| Hashtags | 0.231±0.000$^{\triangledown}$ | 0.181±0.008$^{\blacktriangledown}$ | 0.224±0.001$^{\triangledown}$ |
| Hashtags-T | 0.227±0.001$^{\triangledown}$ | 0.193±0.004$^{\blacktriangledown}$ | 0.227±0.001$^{\triangledown}$ |
| Hashtags-TI | 0.234±0.001$^{\triangledown}$ | 0.172±0.005$^{\blacktriangledown}$ | 0.233±0.001$^{\triangledown}$ |
| Hashtags-TI-X20 | 0.233±0.001$^{\triangledown}$ | 0.143±0.007$^{\blacktriangledown}$ | 0.210±0.005$^{\blacktriangledown}$ |
| Hashtags-TI-X40 | 0.233±0.000$^{\triangledown}$ | 0.158±0.005$^{\blacktriangledown}$ | 0.230±0.001$^{\triangledown}$ |
| Hashtags-TI-X60 | 0.233±0.000 | 0.196±0.002$^{\blacktriangledown}$ | 0.234±0.003$^{\triangledown}$ |
| Hashtags-TI-X80 | 0.230±0.000$^{\triangledown}$ | 0.189±0.004$^{\blacktriangledown}$ | 0.230±0.004$^{\triangledown}$ |
| Hashtags-TI-QL1 | 0.106±0.060$^{\blacktriangledown}$ | 0.026±0.008$^{\blacktriangledown}$ | 0.165±0.048$^{\blacktriangledown}$ |
| Hashtags-TI-QL2 | 0.229±0.001$^{\triangledown}$ | 0.123±0.011$^{\blacktriangledown}$ | 0.233±0.021$^{\blacktriangledown}$ |
| Hashtags-TI-QL3 | 0.234±0.000 | 0.106±0.004$^{\blacktriangledown}$ | 0.231±0.015$^{\blacktriangledown}$ |
| Hashtags-TI-QL5 | 0.235±0.000 | 0.118±0.004$^{\blacktriangledown}$ | 0.233±0.003$^{\triangledown}$ |
| Hashtags-TI-QL20 | 0.231±0.000$^{\triangledown}$ | 0.162±0.003$^{\blacktriangledown}$ | 0.228±0.001$^{\triangledown}$ |

Combining the top two observations leads us to conclude that our approach to constructing pseudo test collections works. We can successfully use pseudo test collections, as long as we find appropriate surrogate relevance labels. Why are these findings important? To train learning to rank methods on microblog retrieval tasks we do not have to invest in manual annotations but can use hashtags for creating training examples. Pseudo test collections can also be used successfully for tuning parameters of retrieval models.

The third observation is that the differences between our pseudo test collections are limited. More advanced methods for selecting tweets and hashtags result in performance that is only sporadically better than the naive baseline method, which treats all tweets and hashtags equally. We can look at this from two angles. (i) Collection construction: We can limit time spent on constructing a smooth and interesting pseudo test collection by substituting more advanced methods (Hashtags-T and -TI) with the naive Hashtags method. Using the naive method is faster and results in a larger collection, with similar results. (ii) Training volume: Investing in obtaining more interesting tweets and hashtags using our more advanced methods substantially reduces the number of queries in our collections. While the naive Hashtags uses over 1,800 queries and 460,000 relevant tweets, the other methods use only 890 (Hashtags-T) and 480 (Hashtags-TI) queries and equally reduced sets of relevant tweets. Training efficiency improves substantially by limiting the number of queries in the collections. In other words, we can choose between spending more time on constructing our collections, while reducing training time, or take a naive collection construction approach that results in larger collections and thus longer training times. A similar observation holds for the editorial collections, which are the smallest collections (50–60 queries), but (supposedly) with the highest quality.

**Table 13:** P30 performance on 2011 and 2012 topics for retrieval models for which MAP was tuned on 2011 topics, ordered by 2012 performance. The only parameter free retrieval model, DFRee, achieved 0.416 on the 2011 topics, and 0.346 on the 2012 topics.

| Model | 2011 | 2012 |
|---|---|---|
| DFR-FD (Terrier) | 0.416 | 0.351 |
| Tf-idf (Terrier) | 0.397 | 0.348 |
| PL2 (Terrier) | 0.406 | 0.336 |
| PRF (Terrier) | 0.391 | 0.335 |
| Tf-idf (Indri) | 0.413 | 0.331 |
| Okapi (Indri) | 0.409 | 0.329 |
| LM (Indri) | 0.404 | 0.325 |
| BUW (Indri) | 0.128 | 0.154 |
| BOW (Indri) | 0.094 | 0.134 |

**Table 14:** MAP performance on 2011 and 2012 topics for retrieval models for which MAP was tuned on 2011 topics, ordered by 2012 performance. The only parameter free retrieval model, DFRee, achieved 0.351 on the 2011 topics, and 0.216 on the 2012 topics.

| Model | 2011 | 2012 |
|---|---|---|
| DFR-FD (Terrier) | 0.352 | 0.220 |
| Tf-idf (Terrier) | 0.352 | 0.215 |
| PL2 (Terrier) | 0.348 | 0.209 |
| PRF (Terrier) | 0.335 | 0.201 |
| Tf-idf (Indri) | 0.357 | 0.194 |
| Okapi (Indri) | 0.354 | 0.191 |
| LM (Indri) | 0.346 | 0.188 |
| BUW (Indri) | 0.075 | 0.069 |
| BOW (Indri) | 0.060 | 0.061 |

Summarizing, we have studied the use of pseudo test collections for training and tuning LTR systems for microblog retrieval. We use hashtags as surrogate relevance labels and generate queries from tweets that contain the particular hashtag. These pseudo test collections are then used for (1) tuning parameters of various retrieval models, and (2) training learning to rank methods for microblog retrieval. We explore three ways of constructing pseudo test collections, (i) a naive method that treats all tweets and hashtags equally, (ii) a method that takes timestamps into account, and (iii) a method that uses timestamps and selects only interesting microblog posts. We compare their performance to those of a randomly generated pseudo test collection and two editorial collections.

Our pseudo test collections have high correlation with the editorial collections in the parameter tuning phase, whereas the random collection has a significantly lower correlation. In the LTR phase we find that in most cases our collections do not perform significantly worse than the editorial collections, while the random collection does perform significantly worse.

Looking forward, we are interested in training on a mixture of editorial and generated ground truth. Our work is related to creating ground truth in a semi-supervised way and we also aim to further explore this relation.

## Acknowledgements

This research was partially supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements nr 258191 (PROMISE Network of Excellence) and 288-024 (LiMoSINe project), the Netherlands Organisation for Scientific Research (NWO) under project nrs 640.004.802, 727.011.005, 612.001.116, HOR-11-10, the Center for Creation, Content and

Technology (CCCT), the BILAND project funded by the CLARIN-nl 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), and the Netherlands eScience Center under project number 027.012.105.

# 9. REFERENCES

[1] G. Amati and C. van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, 2002.

[2] N. Asadi, D. Metzler, T. Elsayed, and J. Lin. Pseudo test collections for learning web search ranking functions. In *SIGIR '11*, pages 1073–1082, 2011.

[3] L. Azzopardi, M. de Rijke, and K. Balog. Building simulated queries for known-item topics: an analysis using six european languages. In *SIGIR '07*, pages 455–462, 2007.

[4] R. Berendsen, M. Tsagias, M. de Rijke, and E. Meij. Generating pseudo test collections for learning to rank scientific articles. In *CLEF '12*, 2012.

[5] M. Bron, E. Meij, M. Peetz, M. Tsagkias, and M. de Rijke. Team COMMIT at TREC 2011. In *TREC 2011*, 2011.

[6] S. Carter, W. Weerkamp, and E. Tsagkias. Microblog language identification. *Language Resources and Evaluation Journal*, 47(1), 2013.

[7] B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *SIGIR '06*, pages 268–275, 2006.

[8] B. Carterette, V. Pavlu, E. Kanoulas, J. A. Aslam, and J. Allan. Evaluation over thousands of queries. In *SIGIR '08*, pages 651–658, 2008.

[9] S. Clinchant and E. Gaussier. Bridging language modeling and divergence from randomness models: A log-logistic model for ir. In *ECIR '09*, pages 54–65, 2009.

[10] B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison Wesley, 2009.

[11] S. Cronen-Townsend and W. B. Croft. Quantifying query ambiguity. In *HLT '02*, pages 104–109, 2002.

[12] M. Efron. Information search and retrieval in microblogs. *J. Am. Soc. Inf. Sci. Technol.*, 62:996–1008, 2011.

[13] M. Efron and G. Golovchinsky. Estimation methods for ranking recent information. In *SIGIR '11*, pages 495–504, 2011.

[14] K. Hofmann, B. Huurnink, M. Bron, and M. de Rijke. Comparing click-through data to purchase decisions for retrieval evaluation. In *SIGIR '10*, pages 761–762, 2010.

[15] B. Huurnink, K. Hofmann, and M. de Rijke. Simulating searches from transaction logs. In *SIGIR 2010 Workshop on the Simulation of Interaction*, 2010.

[16] B. Huurnink, K. Hofmann, M. de Rijke, and M. Bron. Validating query simulators: an experiment using commercial searches and purchases. In *CLEF '10*, pages 40–51, 2010.

[17] T. Joachims. Training linear SVMs in linear time. In *KDD '06*, pages 217–226, 2006.

[18] J. Kim and W. B. Croft. Retrieval experiments using pseudo-desktop collections. In *CIKM '09*, pages 1297–1306, 2009.

[19] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *WWW '10*, pages 591–600, 2010.

[20] G. G. Lee et al. SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *TREC 2001*, pages 442–451, 2001.

[21] J. Lin, C. Macdonald, I. Ounis, and I. Soboroff. Overview of the TREC 2011 Microblog track. In *TREC 2011*, 2012.

[22] T.-Y. Liu. Learning to rank for information retrieval. *Found. and Trends in Inf. Retr.*, 3:225–331, 2009.

[23] C. Lundquist, D. Grossman, and O. Frieder. Improving relevance feedback in the vector space model. In *CIKM '97*, pages 16–23, 1997.

[24] C. Macdonald, R. Santos, and I. Ounis. The whens and hows of learning to rank for web search. *Inf. Retr.*, pages 1–45, 2012.

[25] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[26] K. Massoudi, E. Tsagkias, M. de Rijke, and W. Weerkamp. Incorporating query expansion and quality indicators in searching microblog posts. In *ECIR '11*, pages 362–367, 2011.

[27] D. Metzler. *A Feature-Centric View of Information Retrieval*. Springer, 2011.

[28] G. Mishne and M. de Rijke. A study of blog search. In *ECIR '06*, pages 289–301, 2006.

[29] A. Mohan, Z. Chen, and K. Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. *J. Mach. Learn. Res., Workshop and Conf. Proc.*, 14:77–89, 2011.

[30] N. Naveed, T. Gottron, J. Kunegis, and A. C. Alhadi. Searching microblogs: coping with sparsity and document quality. In *CIKM '11*, pages 183–188, 2011.

[31] J. Peng, C. Macdonald, B. He, V. Plachouras, and I. Ounis. Incorporating term dependency in the DFR framework. In *SIGIR '07*, pages 843–844, 2007.

[32] S. Rajput, V. Pavlu, P. B. Golbus, and J. A. Aslam. A nugget-based test collection construction paradigm. In *CIKM '11*, pages 1945–1948, 2011.

[33] S. Robertson and K. Jones. Simple, proven approaches to text retrieval. Technical report, U. Cambridge, 1997.

[34] S. Robertson and H. Zaragoza. On rank-based effectiveness measures and optimization. *Inf. Retr.*, 10(3):321–339, 2007.

[35] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *TREC '94*, pages 109–109, 1995.

[36] D. Sculley. Large scale learning to rank. In *NIPS Workshop on Advances in Ranking*, 2009.

[37] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *ICML '12*, pages 807–814, 2012.

[38] J. Tague and M. Nelson. Simulation of user judgments in bibliographic retrieval systems. In *SIGIR '81*, pages 66–71, 1981.

[39] J. Tague, M. Nelson, and H. Wu. Problems in the simulation of bibliographic retrieval systems. In *SIGIR '80*, pages 236–255, 1980.

[40] J. Teevan, D. Ramage, and M. R. Morris. #twittersearch: a comparison of microblog search and web search. In *WSDM '11*, pages 35–44, 2011.

[41] W. Weerkamp and M. de Rijke. Credibility-based reranking for blog post retrieval. *Inf. Retr.*, 15(3–4):243–277, 2012.

[42] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01*, pages 334–342, 2001.