Using the Past To Score the Present: Extending Term Weighting Models Through Revision History Analysis

Ablimit Aji Yu Wang
Emory University
{aaji,yu.wang}@emory.edu

Eugene Agichtein
Emory University
eugene@mathcs.emory.edu

Evgeniy Gabrilovich
Yahoo! Research
gabr@yahoo-inc.com

ABSTRACT

The generative process underlies many information retrieval models, notably statistical language models. Yet these models only examine one (current) version of the document, effectively ignoring the actual document generation process. We posit that a considerable amount of information is encoded in the document authoring process, and this information is complementary to the word occurrence statistics upon which most modern retrieval models are based. We propose a new term weighting model, Revision History Analysis (RHA), which uses the revision history of a document (e.g., the edit history of a page in Wikipedia) to redefine term frequency—a key indicator of document topic/relevance for many retrieval models and text processing tasks. We then apply RHA to document ranking by extending two state-of-the-art text retrieval models, namely, BM25 and the generative statistical language model (LM). To the best of our knowledge, our paper is the first attempt to directly incorporate document authoring history into retrieval models. Empirical results show that RHA provides consistent improvements for state-of-the-art retrieval models, using standard retrieval tasks and benchmarks.

Categories and Subject Descriptors

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

General Terms

Algorithms,Theory

Keywords

Term Weighting, Retrieval Models, Collaboratively Generated Content

1. INTRODUCTION

Modern information retrieval systems employ ranking models, where the documents are represented as vectors of terms, which could be either individual words or more sophisticated features. Following the initial feature construction and selection, the notion of term weighting becomes central in these models, as it prescribes the way to compute the importance of a term in a given document. Multiple term weighting approaches have been proposed to date, yet the majority of them are based on the statistics of word occurrence in various corpora. Consequently, it is natural to ask, whether there exist substantially different sources of knowledge about term importance, which are complementary to these frequency-based approaches.

In this paper, we propose and evaluate an approach that relies on a novel source of such knowledge, namely, the revision history of a document. Many information retrieval models, notably statistical language models, assume a generative process of document creation, whereas the terms are chosen to be included in the document according to their importance to the chosen document topic(s), previously chosen terms, and other factors that vary by model. Yet these models only examine one (final) version of the document to be retrieved, effectively ignoring the actual document generation process, even when it is available.

Due to the recent proliferation of collaboratively generated content such as Wikipedia, and other *versioned* content, the document generation process is often directly observable by tracing the edit history of the document. From a stub, a short document grows into a full document, over hundreds and often thousands of incremental edits, through the efforts of many editors. Each step adds or removes content, aspects of a topic, or modifies the emphasis of the document to reflect current events. We posit that a considerable amount of information is encoded in this document authoring process, and this information is complementary to the word occurrence statistics upon which most retrieval models are based.

Prior research explored several alternative directions for term weighting. One family of methods exploits the document structure, and assigns different weights to terms that occur in different parts of the document. Other approaches rely on exogenous statistical information such as the number of times the term occurs in the title of a Wikipedia document or in the query log of a Web search engine, as well as click and browsing statistics on the Web. Yet other approaches employ cluster-based language models or content aggregation across the Web graph. To the best of our knowledge, however, prior studies have not examined the very process of document creation as a source of knowledge about term importance.

We propose a new term weighting model, Revision History Analysis (RHA), which uses the revision history of a document (e.g., the edit history of a page in Wikipedia) to redefine term frequency—a key indicator of document topic/relevance for many retrieval models and text processing tasks. We combine term importance information distilled from the revision history with the conventional statistical approaches, in order to obtain a better characterization of terms' true importance in a document. We then apply RHA to doc-

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

ument ranking by extending two state-of-the-art text retrieval models, namely, BM25 and the generative statistical language model. We believe our paper is the first attempt to directly incorporate document authoring history into retrieval models.

The main contributions of this paper are threefold. First, we propose a novel method, Revision History Analysis (RHA), which examines the edit history of a document to estimate the relative importance of document terms. The *process* of document authoring over time incorporates a significant amount of human editors' knowledge about the world, as well as their judgment of the relative importance of terms for a given topic, hence this source of information is complementary to the existing statistical approaches. Second, we apply RHA to augment two state-of-the-art IR models, namely, BM25 and a statistical language model, which allows us to refine the term weighting computation for ad-hoc information retrieval. Finally, the results of our empirical evaluation show that RHA provides consistent improvements for both BM25 and language model-based retrieval models on standard retrieval tasks and benchmarks. Our work indicates a promising new direction in searching collaboratively generated content.

2. REVISION HISTORY ANALYSIS

This section introduces our general approach and algorithms for revision history analysis. In this paper, we consider the historical revisions of a document as a linear sequence of edits, ignoring special cases of such as reverting revisions to recover from vandalism.

Thus, a page editor modifies a document by adding relevant information or deleting non-relevant information from the previous version of this document, and generates a new version. This process suggests that relevant terms for a web document will frequently appear in most revisions and are rarely deleted. The frequency of these important terms is likely to grow along with the growth of the document length. On the other hand, non-relevant terms may exist in some revisions incidentally, but will be removed by editors in subsequent revisions.

The main observation of this paper, which we attempt to capture in our models, is that the importance of a term in a document can be measured by analyzing the revision history. Section 2.1 mathematically describes this intuition by starting with a simple linear growth model. However, we find that this global model is insufficient, as documents often evolve in *bursts*, when a document undergoes a series or rapid and/or substantial revisions over a short period of time, causing a term to suddenly become more important for the document late in that document's lifetime. Thus, Section 2.2 introduces the burst-based measurement of term relevance.

Our hypothesis is that the term weight for a versioned document should incorporate the term frequency in both the historical versions of the document, and in the latest (current) version of the document. For documents that grow incrementally (that is, following a steady expansion process), this model is sufficient, and is captured by our "global" model. However, some documents undergo series of dramatic changes, as the document is expanded or revised to reflect news events or significant bursts of editing effort, requiring our model to account for such significant changes in the document content. Thus, our final RHA model of term frequency incorporates the "global" term growth (Section 2.1), the "bursty" document generation model (Section 2.2), and the final (latest) version of the document at the time of indexing.

2.1 Global Revision History Analysis

We now introduce our first (and simplest) RHA model, which assumes that a document grows steadily over time.

Consider a document d from a versioned corpus D (e.g., Wikipedia), and $V = \{v_1, v_2, ..., v_n\}$ to be the revision history of d. The number of revisions of document d is n. The latest revision of document d is designated to be the latest document snapshot, $v_n \triangleq d$. Finally, let $c(t, d)$ be the frequency of term t in d.

We can now introduce a term weight according to the RHA global model, $TF_{global}(t, d)$, that would capture the appearance of t across the sequence of document versions. Intuitively, we wish to support the varying term importance across revisions, for example, to capture the importance of the few original terms used to describe a concept in Wikipedia, compared to terms added later in the document's "lifespan". Specifically, we define the new term weight as:

$$TF_{global}(t, d) = \sum_{j=1}^{n} \frac{c(t, v_j)}{j^{\alpha}}, \quad (1)$$

where j is the counter enumerating all revisions of document d from the first revision ($j = 1$), to the last revision ($j = n$). The raw frequency of term t in revision j is indicated by $c(t, v_j)$, is modified using the *decay factor* j^{α}, where α controls the speed of the decay. This decay factor, j^{α} adjusts the relative term weight across the multiple revisions to reflect the importance of term appears in different stages of the document evolution. For example, when $\alpha > 0$, the weight of a term will decrease in later revisions, to reflect the importance of a term appearing early in the document lifetime. In contrast, when $\alpha < 0$, the decay factor rewards the terms appearing in the latter revisions. In our experiments, we found that the optimal value for α was 1.1, implying that the term is more important if it appears early in the revision history of a document.

2.2 Revision History Burst Analysis

Documents can undergo intensive editing or massive content change when the popularity of a document increases, or when related events happen, which are immediately described on the document. We call such situations *bursts*, and extend the "global" decay model described above to capture these kinds of document evolution. These bursts are significant since the topic of a document may be different after the burst. For example, the content of a document may be updated to reflect the latest news, and as a result the topic of the document can shift over time, as the news evolve.

For example, consider a Wikipedia page devoted to the movie "Avatar". In the earlier (ca. June 2006) revisions of the page, there was little editing activity and little content, the page simply mentioned that James Cameron would direct the film, which was going to be released in 2009. However, in October 2006, there is a dramatic change to the content as new details about the plot, budget, and development are added. There is another "burst" in December describing the production and more details about filming. However, in the Wikipedia page that describes the meaning of the Hindu concept "Avatar", its etymology and associated deities, the addition of content increment is considerably slower and editing bursts are much less frequent than in the movie-related page. Consequently, in the movie page, term weights are adjusted by incorporating burst history. In what follows, we present the RHA burst model, and we describe how to detect these bursts in Section 2.3.

Recall, that the main assumption underlying the RHA model is that important terms are introduced early in the life of a document. However, a burst "resets" the decay clock for a term, in a way it "renews" the importance of the term. The intuition here is that if the term is still around after a major rewrite of the document content, then this term must be important. Note that a document could have multiple bursts over its revision history, as can be captured naturally in our approach.

Let $B = \{b_1, b_2, ..., b_m\}$ be the set of *burst indicators* for document d, and $m = |B|$ is the number of bursts. The value of b_j is the revision index of the end of the j-th burst of document d. We define the term weighting for the burst model to be the sum of decayed term weights over all the detected bursts. Each burst "boosts" the term weight for a short time, which then decays just like in the global model. Then, the overall term frequency weight of t is defined as:

$$TF_{burst}(t,d) = \sum_{j=1}^{m} \sum_{k=b_j}^{n} \frac{c(t,v_k)}{(k-b_j+1)^{\beta}}, \quad (2)$$

where k is the counter enumerating the revisions after burst b_j for each j. The raw frequency of term t in revision j, $c(t,v_k)$ is divided by the decay factor $(k-b_j+1)^{\beta}$. Thus, when a burst b_j happens, the decay factor for burst b_j will be set[1] to 1, and then the impact of this burst will gradually decrease in subsequent revisions because the decay factor increases with the growth of k. For a document d, Equation 2 calculates the impact of a burst by summing up term frequency with an exponential decay and adding the impacts of m bursts together. In our experiments, the optimal value for β estimated on the training set was found to be 1.1, equal to the value of α introduced in the preceding section.

For more convenient and effective manipulation we can also represent our burst model in a matrix form. Recall that the decay clock will be reset after each burst event - thus contributing a respective decay factor for each subsequent revision. These can be intuitively represented in a *decay matrix* $\mathbf{W}$, where each row i represents a potential burst position, and each column j represents a document revision. Each entry in W is computed as:

$$\mathbf{w}_{i,j} = \begin{cases} \frac{1}{(j-i+1)^{\beta}} & \text{if } i \le j, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the matrix W has the following structure:

$$\mathbf{W} = \begin{pmatrix} 1 & 1/2^{\beta} & 1/3^{\beta} & \dots & 1/n^{\beta} \\ 0 & 1 & 1/2^{\beta} & \dots & 1/(n-1)^{\beta} \\ 0 & 0 & 1 & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & & 1 \end{pmatrix} \quad (3)$$

where β in the the global parameter/exponent of the decay, and the i-th row of $\mathbf{W}$ corresponds to the set of the decay factors for due to the i-th burst in the editing history. If the only burst in the editing history occurred at revision v_i, the decay factors for the subsequent revisions, are stored in the cells $w_{i,i}, w_{i,i+1}, \dots, w_{i,n}$. The corresponding values are $1, 1/2^{\beta}, 1/3^{\beta}, \dots, 1/(n-i+1)^{\beta}$. Note that the matrix W is triangular, since bursts do not affect any revisions prior to a burst - that is, the columns to the left and above of the cell representing the burst event are not affected by the burst.

Of course, multiple rows in W could be associated with a burst, but probably not all rows – resulting in many *potential* burst positions. Thus, we introduce a vector $U = [u_1, u_2, ..., u_n]$ as the *burst indicator* vector that will be used to "filter" the decay matrix W to contain only the true bursts (we discuss how the bursts are detected in the next section). Specifically, each entry in U, u_j, is set to 1 if a burst is detected at revision j, and is set to 0 otherwise.

We now multiply the row vector U and the decay matrix W, resulting in a vector UW. Each entry of UW, uw_j contains the sum of the decay factors, each one set accordingly to the non-zero respective bursts prior to, and including, the j-th revision. For example, consider the case where $U = [1, 0, 1]$, that is, there was a burst detected in both revisions 1 and 3 but not in revision 2. Then, $uw_3 = 1 \cdot 1/3^{\beta} + 0 \cdot 1/2^{\beta} + 1 \cdot 1$, where the first term is the decay factor from the first burst, the second term is 0 since there was no burst in revision 2, and the third term is 1 since the burst is detected in the current revision and this version's contribution is not yet decayed.

Thus, the term weighting of the RHA burst model for a document d can be finally computed as a scalar dot product between the vector UW and the term frequency vector C, where each entry $c(t,v)$ represents the raw frequency of term t in revision v of the document d. Specifically:

$$TF_{burst}(t,d) = UW \cdot \begin{bmatrix} c(t,v_1) \\ c(t,v_2) \\ c(t,v_3) \\ \vdots \\ c(t,v_n) \end{bmatrix} \quad (4)$$

where UW is the product of the burst indicator vector U with the decay matrix W as described above. The resulting modified term frequency value for a term t and document d, $TF_{burst}(t,d)$, combines the *decayed* values of term frequencies of t across all bursts in the edit history of d. For example, consider the case where d had only three revisions, the burst vector is $U = [1, 0, 1]$ as before, and the frequencies of a term t were $[2, 5, 7]^T$ in the respective versions. Then, the combined term frequency $TF_{burst}(t,d) = 1 \cdot 2 + 1/2^{\beta} \cdot 5 + (1/3^{\beta} + 1) \cdot 7$, where the third term in the sum is "boosted" by the burst in revision 3.

Having described the general burst weighting model, we now turn to the task of actually detecting the burst events.

2.3 Edit History Burst Detection

Documents evolve at different rates and may exhibit a variety of editing activity patterns (as captured by the revision history). For example, as news events happen, some documents may have to be updated to reflect the change in the real world. Other documents may be steadily updated by editors providing more detail or emphasizing certain topics over others. Some of these changes are incremental and gradual, which leaves the article content relatively stable. However, some of the most important or drastic changes are reflected as "bursts" in the content or revision history (e.g., in cases where a real world event requires significant change to a document). Thus, the change in the content of a document or edit activity divides the document into natural local and global episodes that correspond to the burst. We define an edit history "burst" as either intense editing activity or dramatic content change within a time interval. Thus, we propose *content-based* and *activity-based* burst detection algorithms, and a hybrid *combined* algorithm, as described in the rest of the section.

Content-based burst detection.

We consider the relative content change one of the important features signaling potential bursts. The series of revisions $V = v_1, v_2, ..., v_n$ for document d are ordered by the time they appear in the revision history. For a particular pair of revisions (v_{j-1}, v_j) in this revision sequence, if the amount of content change in this interval is above a threshold α, then we consider j to be the end of a content-based burst event $Burst_c$. More formally,

$$Burst_c(v_j) = \begin{cases} 1 & \text{if } \frac{|v_j| - |v_{j-1}|}{|v_{j-1}|} > \alpha, \\ 0 & \text{otherwise.} \end{cases}$$

[1]Notice that $(k-b_j+1)^{\beta} = 1^{\beta}$ for the first revision after burst b_j.

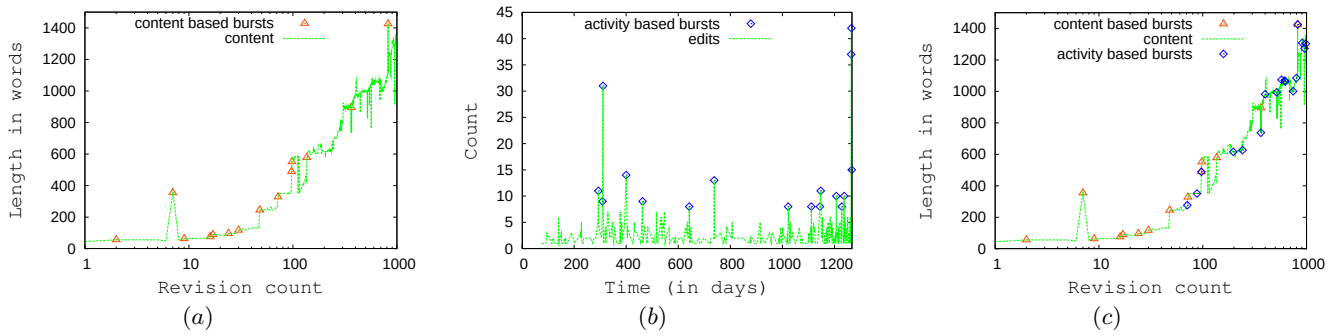

Figure 1: Applying the content-based (a), activity-based (b) and combined (c) burst detection methods to the Wikipedia page "Avatar".

where $|v_j|$ is content length for the j-th revision. The value of threshold α is important: it should not be set too high, as it may miss potential bursts, or too low, as it would cause many false bursts. After development experiments, we set $\alpha = 0.1$ for all subsequent experiments.

Activity-based burst detection.

To model bursts caused by intense editing activity during a certain time period, we consider the edit count in that time period as an important measure signalling bursts for a particular document. That is, we divide the revision history V of a particular document d into *episodes*, where the duration of each episode is Δt; then, an episode is considered *bursty* if the edit count for the episode exceeds "normal" amount of edit activity during the document lifetime. Then, the last revision in that *bursty* episode is selected as the end of the *burst*. More formally, a bursty episode $Burst_a$ is detected as:

$$Burst_a(ep_j) = \begin{cases} 1 & \text{if } |ep_j| > \mu + \sigma, \\ 0 & \text{otherwise.} \end{cases}$$

where μ indicates the average number of edits within an episode and σ is the standard deviation across all the episodes in the document history. For our experiments, we set the episode length to be one day.

Combined burst detection.

Both content-based and activity-based burst detection methods are informative as they capture significant changes in a document. Thus, we combine the two sources of information (content change and the editing activity level). In our experiments we use the simplest combination method of taking the union of the content-based and activity-based bursts. Specifically, the final set of bursts is computed as:

$$Burst(v_j) = \begin{cases} 1 & \text{if } Burst_c(v_j) + Burst_a(v_j) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

The results of different burst detection strategies above to the Wikipedia page titled "Avatar" are illustrated in Figure 1. Figure 1(a) shows the result of applying the content-based algorithm, Figure 1(b) illustrates the result of the activity-based algorithm, and Figure 1(c) shows the result of combining the two methods. As one can see, the combined model is more comprehensive as it captures both types of significant events in the "life" of a document.

Recall, that in the RHA burst model, $Burst(v_j)$ will be used as the j-th entry of burst indicator vector U defined in Section 2.2. This completes the description of the RHA revision history analysis method, and we now turn to incorporating RHA into retrieval models.

3. INCORPORATING RHA IN RETRIEVAL MODELS

This section describes how RHA can be incorporated into two state-of-the-art IR models, namely, BM25 (Section 3.1), as well as into statistical language models (Section 3.2).

3.1 RHA in BM25

We now introduce the way we integrate both the global model and the burst model of RHA into the Okapi BM25 ranking function. The original Okapi BM25 ranking function is defined as:

$$S(Q,d) = \sum_{t \in Q} IDF(t) \cdot \frac{TF(t,d) \cdot (k_1+1)}{TF(t,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})} \quad (5)$$

where $TF(t,d)$ represents term frequency for term t in document d. Inverted document frequency $IDF(t)$ for term t is calculated as:

$$IDF(t) = log \frac{N - n(t) + 0.5}{n(t) + 0.5} \quad (6)$$

where N is the number of documents in the collection, $N = |D|$, and $n(t)$ is number of documents containing term t.

BM25+RHA: We now formally define the modified BM25 model, *BM25+RHA*, that incorporates the RHA term weighting. The main change is that we replace the term frequency in BM25 with the *modified* term frequency TF_{RHA}, which is the mixture of the global and the burst models as well as the standard term frequency computed from the latest revision of the document. Specifically, TF_{RHA} is computed as:

$$TF_{RHA}(t,d) = \lambda_1 TF_{global}(t,d) + \lambda_2 TF_{burst}(t,d) + \lambda_3 TF(t,d) \quad (7)$$

where $TF_{global}(t,d)$ is defined in Equation 1, and $TF_{burst}(t,d)$ is defined in Equation 2. The final TF_{RHA} value is thus a linear combination of the global and burst components, weighted so that $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

Using the modified term frequency TF_{RHA}, the BM25 ranking function with RHA term weighting is redefined as:

$$S_{RHA}(Q,d) = \sum_{t \in Q} IDF(t) \cdot \frac{TF_{RHA}(t,d) \cdot (k_1+1)}{TF_{RHA}(t,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})} \quad (8)$$

The BM25 parameters are set to $k_1 = 1, b = 0.5$, which are the default values in the Lemur Toolkit. While these parameters can also be optimized for the RHA modification, we decided to keep the standard Lemur parameters to make our results easier to replicate.

Extending to Multiple Field BM25 Model: Recently, a fielded variant of BM25, called BM25F, has been demonstrated to improve the performance of the BM25 model, by separately weighting the contribution of terms from the different fields in the document [27]. RHA can be naturally incorporated into BM25F by separately computing the TF_{RHA} values for each field of the document (without any additional changes to the above method). Our preliminary experiments with BM25F+RHA model appear promising, and will be further explored in future work.

3.2 RHA for Statistical Language Models

We now show how RHA can be integrated into the language modeling approach for document ranking. Let D be the collection, $P(t|d)$ be the conditional probability of term t being generated by document d, and $P(t|Q)$ be the probability of term t being generated by query Q. We apply Kullback-Leibler divergence as the ranking function, following Zhai and Lafferty [21]. To score a document d w.r.t to a given query Q, we estimate the query language model and document language model, then score the document as follows:

$$S(Q,d) = D(Q||d) = \sum_{t \in V} P(t|Q) \log \frac{P(t|Q)}{P(t|d)} \quad (9)$$

where V is the set of all words in the vocabulary. Thus, the main task of the ranking is to estimate conditional query term probability $P(t|Q)$ and document term probability $P(t|d)$. Generally the document language model estimated with some form of smoothing, with Dirichlet prior smoothing has been showed to be one of the most effective smoothing methods. With Dirichlet prior smoothing, $P(t|d)$ estimated as:

$$P(t|d) = \frac{c(t,d) + \mu P(t|D)}{|d| + \mu} \quad (10)$$

where $c(t,d)$ is the count of term t in document d, μ is a smoothing parameter that is often set empirically, and $P(t|D)$ is the collection term probability which is estimated as $\frac{\sum_{d \in D} c(t,d)}{\sum_{d \in D} |d|}$. The query term probability is estimated as $P(t|Q) = \frac{c(t,Q)}{|Q|}$.

LM+RHA: We now formally define our modified LM model, *LM+RHA*. The main change to the original LM model above is that we redefine the conditional document term probability $P(t|d)$ as $P_{RHA}(t|d)$, which in turn is computed as the linear combination of the probability derived from RHA and that from the latest version of the document:

$$P_{RHA}(t|d) = \lambda_1 P_{global}(t|d) + \lambda_2 P_{burst}(t|d) + \lambda_3 P(t|d) \quad (11)$$

where $P_{global}(t|d)$ is the probability of term t generated by the revision history of document d, and $P_{burst}(t|d)$ is the probability of term t generated by the bursts within the revision history of document d. Specifically, $P_{global}(t|d)$ is computed as:

$$P_{global}(t|d) = \frac{\sum_{j=1}^{n} \frac{c(t,v_j)}{j^{\alpha}}}{\sum_{t' \in d} \sum_{j=1}^{n} \frac{c(t',v_j)}{j^{\alpha}}} \quad (12)$$

obtained by normalizing $TF_{global}(t,d)$ with the sum of all the term frequencies of t across all the revisions of the document.

$P_{burst}(t|d)$ is defined as:

$$P_{burst}(t|d) = \frac{\sum_{j=1}^{m} \sum_{k=b_j}^{n} \frac{c(t,v_k)}{(k-b_j+1)^{\beta}}}{\sum_{t' \in d} \sum_{j=1}^{m} \sum_{k=b_j}^{n} \frac{c(t',v_k)}{(k-b_j+1)^{\beta}}} \quad (13)$$

obtained by by normalizing $TF_{burst}(t,d)$ with the sum of burst weights across all bursts in the document edit history.

Finally, the third term of Equation 11, $P(t|d)$, describes the probability of a term t generated by the latest version of the document d with Dirichlet prior smoothing, computed using Equation 10. As before, λ_1, λ_2, and λ_3 are tunable parameters that are scaled so that $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Having described how RHA can be incorporated into two example retrieval models, we now turn to the specifics of how RHA can be incorporated into a working retrieval system.

4. SYSTEM IMPLEMENTATION

This section describes the general architecture and relevant components of our system: indexing, retrieval, and ranking. To experimentally evaluate our system, we built a prototype Wikipedia search engine, by incorporating RHA into state-of-the-art retrieval models, as described above. Figure 2 depicts the main components of the system. In our current implementation, we used the latest versions of the Wikipedia articles as the retrieved documents, and the Lemur Toolkit for implementing the retrieval models.

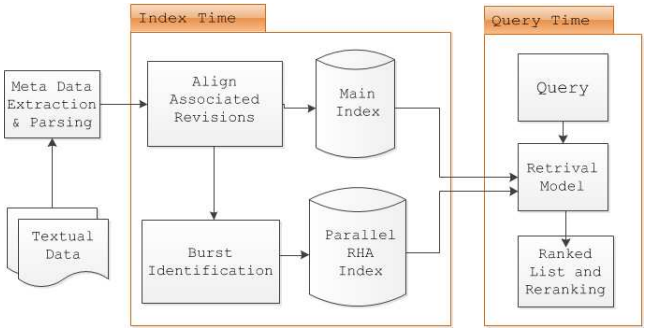

Figure 2: An example system implementation using RHA and associated indexes to facilitate searching of versioned content.

4.1 Wikipedia Parsing

Our Wikipedia parser ignores all the hyperlinks and treats the anchor text as a regular part of the document. Text inherited from templates is included as regular document text. We keep track of the meta-data required for RHA analysis, such as the date the document was created or edited, content change with previous revision, whether it is a minor version, and other editing history information. Other Wikipedia-specific meta-data such as references, infoboxes, tables, and category labels are ignored.

4.2 Indexing

To efficiently rank the retrieved results at query time, we built separate indices for the revision history the latest version of the document. When indexing the latest revisions of the documents, the term count and document length are calculated directly, and this index is used as the corpus for the retrieval task. For the revision history, we build another, augmented, index to support the

RHA-enabled retrieval. Each revision of a document is indexed as a separate document. This is useful for flexible revision history scoring and building different retrieval models based on revision history. The meta data of revisions, such as the birth time of each revision, is stored separately from the index, for efficiency.

4.3 Retrieval

During retrieval, our system initially operates just like a regular information retrieval system: given a query, our system will retrieve potential list of documents from the latest revisions index for scoring, and passes this initial list to the RHA module. Then, for each candidate document, the document revision history is retrieved. At the same time, the system fetches the burst indicator values, and calculates the term frequencies for global and local weighting. For efficiency considerations, global statistics for terms and other parameters necessary for smoothing are pre-computed. In the future, we plan to further optimize the retrieval efficiency by pre-computing the RHA term weights for all documents, instead of performing RHA analysis and re-ranking at retrieval time.

As the underlying search system we utilized the state of the art retrieval models for ranking retrieved results with adjusted term weights. It has been shown in many INEX Ad-Hoc Track experiments [18] that Okapi BM25 is one of the most robust and effective ranking models. One such system is Topx2.0 [28], which has been provided as reference runs for participants, and used as our baseline for the experiments in Section 6. We also used the language modeling based retrieval models, as in many TREC tasks and web retrieval tasks language modeling has consistently performed better than other known retrieval models. In all our experiments we used distance-based likelihood variants of the language model ranking (provided by the Lemur toolkit), and our model is smoothed with Dirichlet prior smoothing, with the prior value of $\mu = 1000$.

5. EXPERIMENTAL SETUP

We now describe the evaluation metrics (Section 5.1) used to compare the retrieval methods (Section 5.2) on two standard benchmark datasets used in this study (Section 5.3).

5.1 Evaluation Metrics

A common practice of evaluating IR systems is to perform pooling of judged documents for each query/topic [15]. However, this approach assumes most relevant documents have been judged, and hence considers non-judged documents to be irrelevant. When each query has numerous relevant documents, it can happen that the top N retrieved documents in a new run contain a single judged document or even none at all. We address this problem in two different ways. First, Buckley and Voorhees [6] have introduced a new evaluation metric, *bpref*, which allows to overlook non-judged documents and does not require to consider them to be irrelevant (the metric is computed by analyzing the relative rankings of the relevant and irrelevant documents). Second, we compute the standard metrics such as Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG) only for the documents for which we have judgments. Specifically, the metrics are computed as follows:

- **bpref**: Let R denote the number of judged relevant documents and let N denote judged nonrelevant documents. Let D denote a relevant retrieved document, and let $\bar{D}$ denote a member of the first R judged non-relevant documents as retrieved by the system. Then, bpref is computed as:

$$bpref = \frac{1}{R} \sum_{D} (1 - \frac{|\bar{D} \text{ ranked higher than } D|}{min(R, N)})$$

- **MAP**: Mean Average Precision is computed as:

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk})$$

  where Q is the set of queries, and m_j is the set of ranked results retrieved for a query j. The average precision for each query j is computed by adding the precision after each relevant result R_{jk} is retrieved.
- **NDCG**: Normalized Discounted Cumulative Gain metric, introduced by Jarvelin and Kekalainen [17], extends MAP to graded relevance judgements and places higher weight on relevance in the top results, while discounting the gain of relevant documents retrieved in the lower result positions. The value is then normalized by the best possible discounted gain that could be obtained by perfectly re-ranking the retrieved results.
- **R-Precision**: the number of relevant documents in the collection can vary significantly across queries. R-Precision is computed as precision of the top R documents, where R is the number of relevant documents in the collection for that query. R-Precision provides a useful and complementary information to other metrics above.

5.2 Methods Compared

We compare the following ranking methods experimentally:

- **BM25**: standard implementation of BM25 in Lemur, with default parameter values.
- **BM25+RHA**: Our extension of BM25 by incorporating the RHA term frequency as described in Section 3.1.
- **LM**: standard implementation of the unigram language model implemented in Lemur, as described above.
- **LM+RHA**: Our extension of LM by incorporating RHA as described in Section 3.2

5.3 Datasets

For our study we used two different sets of query benchmarks. The first is the collection of topics and relevance judgments from the INEX 2009 Ad Hoc track evaluation [18]. The second is a set of TREC ad-hoc queries, with relevance judgments created manually for this study by volunteers.

INEX data: INEX has become a well established IR evaluation forum for comparing different retrieval approaches for structured document retrieval and ranking. Our corpora and benchmarks make use of the Wikipedia collection developed the recent INEX evaluations [12]. Specifically, we used a susbset of the INEX queries and relevance judgments for 50 topics from the INEX 2008 ad-hoc track, and 64 topics for the INEX 2009 ad-hoc track. The topic titles were used as the queries to retrieve top 1000 documents from the (November 2009) snapshot of Wikipedia. For each retrieved document, at most the first 1000 revisions were used for RHA analysis (some retrieved documents had fewer than 1000 revisions).
TREC data: to test the generality and robustness of our system, we used an additional query set of 68 topics randomly sampled

from the TREC ad-hoc track. Topic titles were used as queries to retrieve top 100 documents for each query from the (November 2009) snapshot of Wikipedia, using the BM25 baseline. For each retrieved document, at most the first 1000 revisions were used for RHA analysis (some retrieved documents had fewer than 1000 revisions). For these queries, human relevance labels from volunteers were obtained, by pooling the top 10 results for each of the retrieval methods in Section 5.2.

6. EXPERIMENTAL RESULTS

We now present the experimental results of incorporating RHA into BM25 and LM models. Specifically, we first describe the parameter tuning for RHA to optimize the relative importance of the edit history weights compared to the weights derived from the latest version of the document. We then report results on the INEX and TREC queries over the datasets described above. We conclude this section with error analysis and case studies, to gain better intuition about performance of RHA, and suggest directions for future improvement.

6.1 System Tuning

Parameters λ_1, λ_2, and λ_3 (importance of the final version of the document, global history, and burst history, respectively) were tuned by maximizing the bpref metric on the INEX queries and their relevance judgments.

Table 1 reports results of evaluating RHA on the INEX 2009 queries and judgments, with the best weights chosen for λ_1, λ_2, and λ_3 parameters using exhaustive search. RHA provides consistent improvement over the baseline models on all metrics, with roughly 5% relative improvement in bpref and R-Precision, and moderate improvement of 2% on the MAP metric. The optimal parameter weights for the BM25+RHA model were $\lambda_1 = 0.3$, $\lambda_2 = 0.4$, and $\lambda_3 = 0.3$. For the LM+RHA model the optimal parameter values were $\lambda_1 = 0.3$, $\lambda_2 = 0.2$, and $\lambda_3 = 0.5$.

Model	*bpref*	*MAP*	*R-Precision*
BM25	0.354	0.354	0.314
BM25+RHA	**0.375** (+5.93%)	0.360 (+1.69%)	0.337 (+7.32%)
LM	0.357	0.370	0.348
LM+RHA	0.372 (+4.20%)	**0.378** (+2.16%)	**0.359** (+3.16%)

Table 1: Retrieval performance improvements when incorporating RHA into BM25 and LM models (INEX 2009 query set). Bolded entries indicate the best result for each performance metric across all retrieval models.

6.2 Results on the INEX data

We now simulate a more realistic retrieval setting where the tuning of the parameters is performed on a separate training set to avoid overfitting. Specifically, we perform 5-fold cross validation separately on both the INEX 2008 and INEX 2009 datasets. For each round of the cross validation, we first tuned parameters according to the best value of bpref on the training set, then applied the well-tuned RHA on the complementary validation set.

Table 2 and Table 3 reports cross-validation results on INEX 2008 and INEX 2009 query sets. The results show that RHA consistently outperforms baseline retrieval methods on bpref and MAP metrics. For INEX 2008 queries, LM+RHA outperformed the baseline LM model with 8.7% relative improvement on the bpref metric, and 4.9% improvement on the MAP metric. We conjecture that the slight degradation of *R-Precision* of BM25+RHA could be due to the optimization process that maximized the *bpref* metric. Interestingly, the improvement for the INEX 2009 is not as large as it is for INEX 2008 queries, but is still significant on bpref and R-Precision metrics. We conjecture that the effects could be explained by the differences in the queries between the two datasets, as we analyze in more detail at the end of this section. But, in general, identifying what kind of queries could benefit from RHA is an interesting future research direction.

Model	*bpref*	*MAP*	*R-Precision*
BM25	0.307	0.281	0.324
BM25+RHA	0.312 (+1.63%)	0.291 (+3.56%)	0.320 (-1.23%)
LM	0.311	0.284	0.330
LM+RHA	**0.338** (+8.68%)	**0.298** (+4.93%)	**0.332** (+0.61%)

Table 2: Retrieval performance improvements when incorporating RHA into BM25 and LM models (INEX 2008 query set with 5-fold cross validation).

Model	*bpref*	*MAP*	*R-Precision*
BM25	0.354	0.354	0.314
BM25+RHA	0.363 (+2.54%)	0.348 (-1.69%)	0.333 (+6.05%)
LM	0.357	0.370	0.348
LM+RHA	**0.366** (+2.52%)	**0.375** (+1.35%)	**0.352** (+1.15%)

Table 3: Retrieval performance improvements when incorporating RHA into BM25 and LM models (INEX 2009 query set with 5-fold cross validation).

6.3 Results on the TREC data

Table 4 reports the performance results on TREC queries. The improvement due to incorporating RHA into the retrieval models for this benchmark are particularly promising. This is especially true for BM25+RHA model that exhibits 3.65% relative improvement on MAP, 3.47% improvement on NDCG, and 4.39% improvement on bpref compared to the baseline BM25 model (all improvements are statistically significant with $p \leq 0.01$). These results were obtained on a different dataset (from the INEX dataset used for tuning), without re-tuning any parameters. Therefore, these results indicate that RHA is a general and effective method for enhancing IR ranking models.

Model	*bpref*	*MAP*	*NDCG*
BM25	0.524	0.548	0.634
BM25+RHA	**0.547**‡ (+4.39%)	**0.568**‡ (+3.65%)	**0.656**‡ (+3.47%)
LM	0.527	0.556	0.645
LM+RHA	0.532 (+0.95%)	0.567 (+1.98%)	0.653 (+1.24%)

Table 4: Retrieval performance improvements for TREC queries, ‡ indicates significant differences at $p = 0.01$ level with two-tailed paired t-test.

6.4 Error Analysis and Case Studies

We observed that for some queries, the improvements are quite large, whereas for others, RHA does not have a noticeable effect,

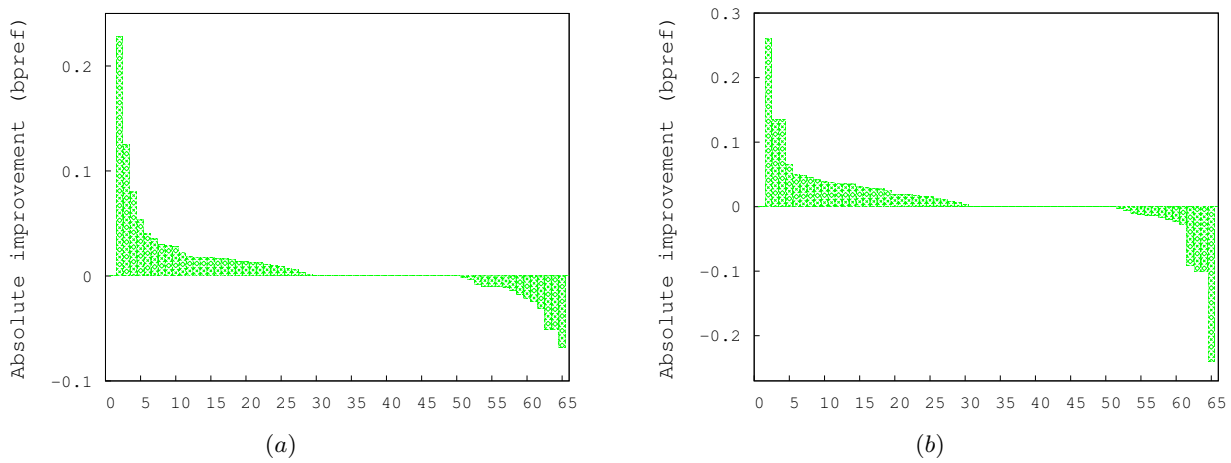

Figure 3: Performance improvements on bpref for BM25+RHA (a) and LM+RHA (b) retrieval models for individual INEX 2009 queries.

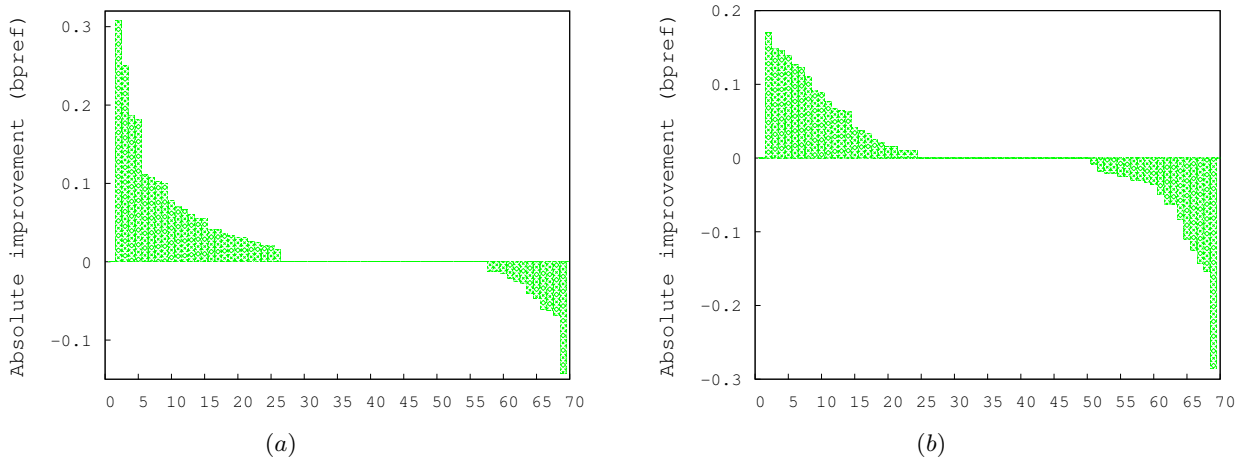

Figure 4: Performance improvement on bpref for BM25+RHA (a) and LM+RHA (b) retrieval models for TREC queries.

and for a small fraction of queries, RHA degrades performance. Figure 3 (a-b) reports the absolute improvement on the bpref metrics for individual INEX 2009 queries, when incorporating RHA into BM25 and LM retrieval models, respectively. RHA on BM25 has a noticeable improvement of bpref on 40% queries, while RHA on LM helps 34% queries. Some queries, such as "circus acts skills" and "olive oil health benefit", gain substantial improvements from RHA. For example, for the query "olive oil health benefit", we saw relative 20% improvement on BM25+RHA model, and relative 11% on LM+RHA model. However, RHA degrades performance on some queries, such as the query "earthquake prediction", which spans multiple topics including geology, natural disasters. We conjecture that the current implementation of RHA is more helpful for focused queries and hurts for ambiguous queries.

We observe a similar pattern on the TREC dataset. Approximately 37% of queries are substantially helped by RHA, and another 18% are degraded by RHA. Just as for the INEX queries, we observe that RHA performs better on focused, unambiguous query topics. One such query is "Metabolism", where the task is to identify documents related to chemical reactions necessary to keep living cells healthy. BM25+RHA performed best on this topic, with the resulting MAP value five times higher than that of the baseline BM25 model. In contrast, RHA appears to degrade performance when the original result set is poor. That is, if the original retrieval result does not contain a sufficient number relevant documents or the query spans multiple topics, RHA-based re-ranking does not improve over the original ranking. One such query is "radio waves and brain cancer" which spans multiple topics. Identifying, and accounting for such cases automatically is a topic of future work.

Interestingly, BM25+RHA performed better for TREC queries, whereas LM+RHA generally performed better for INEX queries.

In summary, our experimental results show consistent improvements due to incorporating RHA into retrieval models across different datasets and metrics. It should be pointed out that these were achieved with little or no tuning of the underlying retrieval model parameters, and both absolute and relative improvements could likely be increased with additional tuning and other relevance feedback techniques. Thus, our results show the value and generality of the RHA model, which, as far as we know, is the first attempt

to incorporate the document authorship process in retrieval models. Next, we place our contribution in the context of prior work.

7. RELATED WORK

There are several main bodies of prior research that are relevant to our study, including various approaches to term weighting and the use of temporal information and other exogenous information for retrieval and ranking.

Term weighting.

Several alternative directions for term weighting have been previously explored in the literature. Zaragoza et al. [27] proposed BM25F, a variant of BM25 [26], which computes term weights depending on which part (field) of the document the term appears in. Similarly, Ogilvie and Callan [24] used structural markup to combine document representations within the language modeling framework. Additional relevant approaches focusing on document structure include the work by Trotman [29] and Wang and Si [30].

Other approaches explored the use of statistical information of term occurrence in datasets other than the target retrieval corpus. Bendersky and Croft [3] identified key concepts in long queries using term frequency in Google's 1 Terabyte N-gram corpus, as well as in a large query log. Subsequently, Lease [22] also studied term weighting for verbose queries within the Markov random field model. Although these studies focused on query-side term weighting, in principle similar approaches could be applied to document-side term weighting as well. Other studies also modified the standard language modeling approach by considering relationships between words in a document [7]. A number of studies also developed new term weighting methods using topic models and cluster-based language models [20, 23, 31]. To the best of our knowledge, however, prior publications have not considered the process of document creation as a source of knowledge about term importance, a direction we explored in this paper.

Exploiting temporal information for IR tasks.

Several prior studies focused on versioned document retrieval models, where the objective is to efficiently access previous versions of the same document [32, 16]. Research on topic detection and tracking (TDT) analyzed the evolution of stories and topics over time [2]. Gabrilovich et al. [13] studied the dynamics of information novelty in evolving news stories. Olston and Pandey [25] introduced the notion of information longevity to devise more sophisticated crawling policies. Gruhl et al. [14] and others [19] analyzed temporal information diffusion in blogosphere, including the temporal patterns in term popularity (in fact, we adapt one of the proposed methods in [14] for detecting significant events in a document edit history).

Two prior studies are arguably the most relevant to our work. Elsas and Dumais [10] studied the dynamics of document content with applications to document ranking. There are several key differences between that work and ours. First, Elsas and Dumais only consider repeated crawls of web documents—as opposed to formally tracked revisions of a Wikipedia page, they consider periodic snapshots, starting from an unknown point in document's life. Second, they do not model term evolution, and instead manipulate three groups of terms defined by their lifespan. Third, we explicitly model "significant" events in the document's lifespan by analyzing the bursts of change activity. Finally, we provide a general term frequency model that can be incorporated into any retrieval model, and not only in LM. More generally, our work could be used to improve the family of probabilistic retrieval models other than BM25 [11]. Efron [8] also used temporal information for determining term weights, yet he considered the change over time of the *entire collection*, rather than of individual documents.

Exploiting Other Metadata for IR.

In addition to temporal information, CGC provides other rich metadata that has been exploited for ranking. For example, user feedback such as votes and author reputation information has been successfully used for searching community question answering archives [4] and for blog search [9]. Other useful metadata shown useful for ranking includes behavioral data such as clickthrough and browsing information (e.g., [1, 5]).

In contrast to previous work, our RHA model combines the global and local time-frame analysis of the edit history, with the local history punctuated by "bursts" in document edit activity. To our knowledge, our paper is the first attempt to directly incorporate document authoring history into retrieval models in a general way.

8. CONCLUSIONS AND FUTURE WORK

This paper introduced a novel term weighting scheme that uses Revision History Analysis (RHA) of the document edit history. Unlikely previous models, RHA directly captures the document authoring process when available, and is particularly valuable for collaboratively generated content, notably Wikipedia documents. RHA can be naturally incorporated into state of the art retrieval models, as we demonstrate by showing consistent improvements that RHA enables for BM25 and LM retrieval models. Other potential applications of RHA include document classification, clustering, and feature selection – as all of these tasks make use of the term frequency information.

A natural next step is to apply RHA to other types of versioned content where authorship and edit history information is available, such as source code repositories, forum threads, and Community Question Answering archives. A complementary direction is to use RHA term weighting as a source of additional external evidence for general web ranking, which we plan to explore in our future work.

Acknowledgments

We gratefully acknowledge support for this research by the HP Labs "Open Innovation" award and by the Yahoo! Faculty Research Engagement awards. We also thank Donald Metzler for pointing us to several relevant bodies of prior work.

9. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proc. of SIGIR*, 2006.

[2] J. Allan. Introduction to topic detection and tracking. In *Topic detection and tracking: event-based information organization*, pages 1–16. Kluwer Academic Publishers, 2002.

[3] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *SIGIR*, pages 491–498, 2008.

[4] J. Bian, Y. Liu, E. Agichtein, and H. Zha. Finding the right facts in the crowd: factoid question answering over social media. In *Proc. of WWW*, 2008.

[5] M. Bilenko and R. W. White. Mining the search trails of surfing crowds: identifying relevant websites from user activity. In *Proc. of WWW*, 2008.

[6] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *SIGIR'04*, 2004.

[7] G. Cao, J. Nie, and J. Bai. Integrating word relationships into language models. In *Proc. of SIGIR*, 2005.

[8] M. Efron. Linear time series models for term weighting in information retrieval. *Journal of the American Society for Information Science and Technology (JASIST)*, 2010.

[9] J. Elsas, J. Arguello, J. Callan, and J. Carbonell. Retrieval and feedback models for blog feed search. In *Proc. of SIGIR*, 2008.

[10] J. Elsas and S. Dumais. Leveraging temporal dynamics of document content in relevance ranking. In *Proc. of WSDM*, 2010.

[11] N. Fuhr and C. Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems (TOIS)*, 9(3):223–248, 1991.

[12] N. Fuhr, J. Kamps, M. Lalmas, S. Malik, and A. Trotman. Overview of the INEX 2007 ad hoc track. *Focused Access to XML Documents*, pages 1–23, 2008.

[13] E. Gabrilovich, S. Dumais, and E. Horvitz. Newsjunkie: Providing personalized newsfeeds via analysis of information novelty. In *WWW*, pages 482–490, 2004.

[14] D. Gruhl, R. V. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *WWW*, 2004.

[15] D. Hawking, N. Craswell, and P. Thistlewaite. Overview of TREC-7 very large collection track. In *TREC-7*, 1998.

[16] J. He, H. Yan, and T. Suel. Compact full-text indexing of versioned document collections. In *CIKM*, pages 415–424, New York, NY, USA, 2009. ACM.

[17] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[18] J. Kamps, S. Geva, and A. Trotman. Analysis of the inex 2009 ad hoc track results. In *INEX*, 2009.

[19] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *Proc. of WWW*, 2003.

[20] O. Kurland and L. Lee. Corpus structure, language models, and ad hoc information retrieval. In *SIGIR*, pages 194–201, 2004.

[21] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 2001.

[22] M. Lease. An improved markov random field model for supporting verbose queries. In *SIGIR*, pages 476–483, 2009.

[23] X. Liu and W. B. Croft. Cluster-based retrieval using language models. *SIGIR*, pages 186–193, 2004.

[24] P. Ogilvie and J. P. Callan. Combining document representations for known-item search. In *SIGIR*, pages 143–150, 2003.

[25] C. Olston and S. Pandey. Recrawl scheduling based on information longevity. In *WWW*, pages 437–446, 2008.

[26] S. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. 3rd TREC*, pages 109–126, 1994.

[27] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *CIKM*, pages 42–49, 2004.

[28] M. Theobald, H. Bast, D. Majumdar, R. Schenkel, and G. Weikum. Topx: efficient and versatile top- query processing for semistructured data. *VLDB J.*, 17(1), 2008.

[29] A. Trotman. Choosing document structure weights. *Information Processing & Management*, 41(2), 2005.

[30] M. Wang and L. Si. Discriminative probabilistic models for passage based retrieval. In *Proc. of SIGIR*, 2008.

[31] X. Wei and W. B. Croft. LDA-based document models for ad-hoc retrieval. *SIGIR*, pages 178–185, 2006.

[32] J. Zhang and T. Suel. Efficient search in large textual collections with redundancy. In *Proc. of WWW*, 2007.