Learning to Rank from Relevance Feedback

Peter Lubell-Doughtie

peter@helioid.com

University of Amsterdam,
Amsterdam, Netherlands

Master of Science Thesis

under the supervision of Maarten van Someren and Katja Hofmann

submitted to the Board of Examiners in partial fulfillment of
the requirements for the degree of Master of Science in Artificial Intelligence

Table of Contents

List of Figures

List of Tables

Summary

A large number of searches involve ambiguous terms, require the retrieval of many documents, or are conducted during multiple interactions with the search system. In these cases user feedback is especially useful for improving search results, although search systems are rarely designed to use this feedback. To address these common scenarios we design a search system that uses novel methods to learn from the user's judgements of search results. By combining the traditional method of query expansion with learning to rank our search system uses the interactive nature of search to improve result ordering, even when there are only a small number of judged documents.

To represent documents for use in learning to rank we build feature spaces from judged documents using query expansion with a varying number of expansion terms, query expansion with different sets of judged documents, and a vector space model based on term frequency. To simulate different methods of requesting document judgements from the user we sample from the highest ranked document, from a set of documents expected to be informative, and from all documents. Our search system uses document judgements as they are provided to create both a query-specific and global profile, which we then use in query expansion and learning to rank.

We find that, excepting certain queries, the query-specific profile outperforms the global profile; although for some queries the global profile performs less reordering, which leads to better scores. We also found inconclusive evidence that the amount of reordering is negatively correlated with performance. We find that, as more documents are judged, all the variations of our method improve in performance; and that random sampling outperforms other sampling strategies for query expansion based features, while term frequency features perform best when sampling from the highest ranked document. We find that, when compared to other feature spaces, learning from constant features outperforms when sampling from informative documents and term frequency features outperform when sampling from the highest ranked documents. We argue that the manner in which feature spaces represent document data interacts with the biases in sampling strategies to contribute to this correlation. Based upon our experimental results we conclude that our learning to rank method improves result ordering beyond that achievable when using solely query expansion.

1 Introduction

Searching for information is an inherently interactive process. In common search systems[1] the user submits a formulation of her request for information, then the system responds with a set of documents, which she evaluates. If the documents sufficiently meet her needs the search is complete, otherwise she submits a refined request or continues searching in some other manner. From the user's perspective this process is necessarily interactive, which begs the question:

> *How can a system that is aware of the interactive nature of search use this interactivity to better serve the user's needs?*

A significant amount of work addresses just this question, e.g. [42, 58, 59, 88, 103], among others. One approach, learning to rank, employs algorithms which use document text, meta-data, retrieval scores, and other data, to learn a function that predicts document relevance. Interest in learning to rank has recently increased and the field has been pushed forward by various machine learning challenges, which provide a feature-rich dataset to participants who are then awarded prizes based upon who can best use the data's features to learn an improved ordering.[2]

The earliest method of learning to rank is relevance feedback. In relevance feedback, judgements on documents returned for an initial query are used to perform a new query. We attempt to improve the effectiveness of relevance feedback by using novel methods that apply the information provided by relevance feedback to build feature vectors for use in learning to rank, as we will describe below.

As learning to rank has become more popular researchers have begun applying it and other methods that exploit interactivity to search within specific domains—such as legal, patent, and academic corpora. Domain specific search is gaining attention and momentum, especially with the introduction of the TREC-Legal[3] and CLEF-IP[4] tracks, which focus respectively on legal document and intellectual property search. These collections are an appropriate place to apply and test learning oriented search systems because users are more willing to exert effort interacting with the search system and the potential benefits of learning from this interaction are immense [22, 63, 85].

Search within these specialized collections is unique in two important ways that relate to our research:

1. recall is more important than precision,
2. queries are contextualized within their domain (they are domain specific).

[1] E.g. Google, Bing, or specialized systems like the primarily legal search engine LexisNexis, `http://www.lexisnexis.com/`.

[2] E.g. the Yahoo Learning to Rank Challenge for ad-hoc search, `http://learningtorankchallenge.yahoo.com/`, which is now complete.

[3] `http://trec-legal.umiacs.umd.edu/`

[4] `http://www.ir-facility.org/clef-ip`

With regards to (1), the specialized collection searcher will usually look through a greater number of documents than the ad-hoc searcher would. Furthermore, this searcher expects many documents will be relevant to her query, which gives her an incentive to provide relevance feedback if it will improve ordering and reduce the total amount of effort expended in her search.

Although specialized collection search is focused on recall, precision is still a factor. Higher precision reduces the number of documents that must be reviewed, the amount of resources expended for review, and the detriments of stopping the review process early—perhaps due to budget or time constraints, an ever present factor in legal and academic search scenarios. By reordering documents as the searcher provides relevance feedback we use the information in judged documents to most effectively improve both precision and recall.

Regarding (2), in domain specific search users often write queries which contain terms-of-art, use non-standard word meanings, and are expressed with specialized terminology.[5] By iteratively learning from relevance feedback our method addresses the challenges of vocabulary and ambiguity in search for specialized collections.

1.1 Research questions

In our research we extend learning to rank by exploring how a search system can use relevance feedback to better rank documents returned for a query. As relevance feedback, we consider judged documents in two contexts: a query-specific context, which includes documents specific to a particular query issued by the user; and a *global* context, which includes documents across all the user's queries and is more representative of her general information needs and interests. With this research we evaluate the following question:

Research Question 1
Does global feedback from queries related to the user's general information need improve a search system's ability to re-rank for new queries?

By iteratively applying our method, as we learn more about users' information needs from their feedback, we answer the following question:

Research Question 2
How is a search system's ability to learn to rank results affected by the number of judged documents?

Judging documents is time-consuming for users and they are often unwilling to expend the necessary effort. When users do judge documents it is important that the search system best uses the information they provide. Perhaps after a certain number of documents have been judged changes in performance plateau.

[5] For example, the TREC 2010 Legal dataset contains queries with the phrase "FAS 140", an opaque reference to Enron corporation's transactions under Financial Accounting Standard 140.

We could then use this information to estimate how many documents for which to request judgements.

If we are able to request document judgements from the user we must select documents to be judged. The dataset we use has a given set of informative judged documents, the *seed* documents, and a larger set of arbitrary judged documents that are used in evaluation, *all* documents. We simulate requesting judgements by using three different strategies to sample from these sets.

The first strategy samples from the seed documents descending from the highest ranked. This is equivalent to asking the user to judge the most relevant documents, when we think their judgement will be informative, and is the manner in which we would expect a real world search system to receive implicit feedback—because users browse documents descending from the highest ranked. This strategy is also similar to *pseudo-relevance feedback*, in which we assume the highest ranked documents are relevant. Because this strategy exploits information in highly ranked documents before exploring lower ranked documents we will refer to it as *exploitative sampling*.

The second strategy randomly samples from the seed documents, and the third randomly samples from all documents. Sampling from the seed documents models the scenario in which there exists a set of documents that we expect will be useful if we receive judgements for them.[6] When sampling from all documents we assume no knowledge of the comparative usefulness of documents in our dataset.

We compare these strategies to answer our next research question:

Research Question 3 *Which sampling strategy most improves retrieval scores when applying learning to rank?*

By comparing exploitative and random sampling we can estimate which strategy produces more useful documents for learning, and for which documents to request judgements in practice. Additionally, by comparing random sampling from the seed set to random sampling from all documents we can discover whether the seed documents provide particularly helpful information, and how much more helpful this information is.

After we have gathered our judged documents, the essential step is learning from them. We use three different feature spaces in learning to rank:

- retrieval scores for different judged document sets,
- retrieval scores for the same judged documents with query expansion using different numbers of expansion terms,
- document term frequency.

We compare retrieval scores for the different feature spaces, as well as for hybrids of these spaces, to answer our final research question:

[6] For example, perhaps we have predicted the amount by which a document's judgement will reduce the version space—the set of hyper-planes that separate documents in an induced feature space [90].

Research Question 4 *Which feature space most improves retrieval scores when applying learning to rank?*

Whichever feature space performs best can be researched more and used in practice. Each of our research questions is answered by varying the design of our search system.

1.2 Methodology overview

We implement a search system that learns from user feedback to build a model with which it reorders documents to better serve the user's information needs. The system iteratively updates its model as judged documents are collected and learns from different feature spaces constructed from these documents. The main components of our search system are:

1. a retrieval function that ranks documents,
2. an expansion function that builds updated queries from judged documents,
3. a learning function that extracts features from judged documents and uses them to reorder.

To build judged document sets we simulate user feedback by sampling based on retrieval score and sampling at random.

Figure 1 presents a high level overview of our search system. The user first

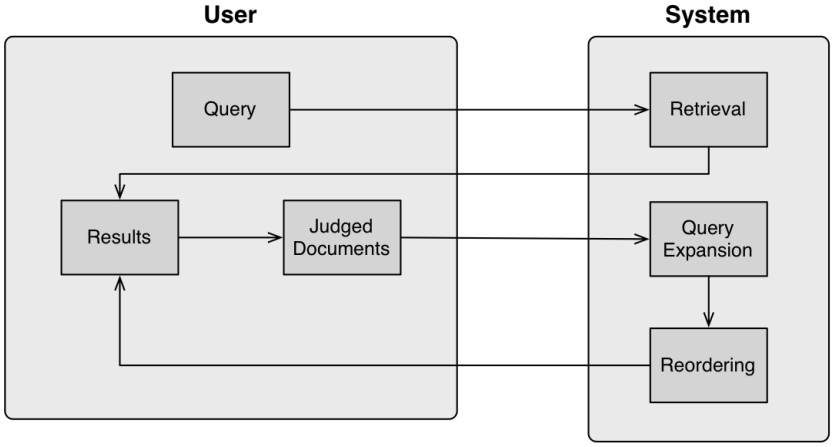


Fig. 1. A high level overview of our search system.

submits a query, with which the system retrieves results. The system then returns these results to the user, who judges a subset of them. Next the system uses these judged documents to expand the query and reorder results through learning to rank. Finally, the reordered results are returned to the user, who is either satisfied and ends the search, or judges more results and continues the retrieval process.

1.3 Contributions

Our research deals primarily with learning to rank based upon relevance feedback. We generically describe our method so that it is applicable to ad-hoc search, personalized search, and other machine learning tasks which use a judged subset of a corpus to order that corpus. The specific contributions of our work include:

- a method of combining query expansion with learning to rank that improves retrieval scores,
- a study of the impact of relevance feedback as we learn more about the user's preferences over time,
- an application of query expansion and learning to rank to the TREC Legal learning task.

With our research questions we contribute a comparison between:

- the benefits of learning with a query-specific profile and a global profile,
- sampling strategies for use with learning to rank,
- ways of building a feature space for use in learning to rank.

1.4 Description of content

We proceed by introducing related work and describing its relationship to our work in Section 2. In Section 3 we describe our method. In Section 4 we present our experiments, the evaluation methodology we will use, and how it serves to answer our research questions in an unbiased setting as well as in a setting that realistically represents a typical searcher's interaction with an information retrieval system.

We present the results of our experiments and compare our results to related work in Section 5. We discuss the implications of these results with respect to our research questions and related research in Section 6. Finally, in Section 7 we conclude and touch upon potential future work based on our contributions.

2 Related work

Our work is at the intersection of research into relevance feedback, search personalization, and learning to rank. Therefore we will primarily discuss related research that overlaps with these subtopics or otherwise directly relates to, or has influenced, our research. Our method is tested on the TREC Legal corpus and shares similarities with previous TREC Legal research in virtue of sharing the same dataset, although the specific search strategies used are not closely related to ours. We will provide an overview of related work from TREC Legal and a brief discussion of its history within TREC.

2.1 Relevance feedback for query expansion

Relevance feedback is concerned with using document judgements—whether these are binary positive/negative assessments or probabilities representative of interest—to produce a better ordering of the returned documents for the current query and future queries. The most common use of relevance feedback is in query expansion, a method which constructs a modified query as a combination of the initial query, a positive contribution from relevant documents, and a negative contribution from non-relevant documents [62, 79]. The most commonly used query expansion method is the Rocchio algorithm.

The Rocchio algorithm describes a well founded way of incorporating document judgements into a new query, but makes limited use of the information in judged documents. These documents can only affect result ordering through the new ordering induced by the expanded query. By using judged documents for query expansion, as well as to build a feature space with which to evaluate document relevance, our research makes better use of the information they contain.

The Rocchio algorithm has been implemented for a variety of models, including vector space models [56] and language models [65, 66, 75, 88]. Research shows [51] that vector space models are suboptimal in some situations, especially when they are based upon term frequency. Li et al. [51] improve scores by combining a term frequency vector space model with language modeling. Methods that combine vector space models and language models require techniques and parameters to perform this combining. A possible improvement is to use a staged method, which begins with query expansion and language model based retrieval, then boosts scores by reordering using a vector space model—this is the approach we take.

For query expansion to be effective it requires a sufficiently large set of judged documents. Query expansion can be ineffective, or detrimental to performance, if the system does not have enough judged documents. Okabe and Yamada [68] address this by using transductive learning to obtain pseudo-relevant documents and then aggregating the predictions of multiple learning trials to sort candidate terms by importance. Our method addresses this by using judged documents in two kinds of learning algorithms: query expansion and learning to rank.

Kurland et al. [49] use a simpler term selection method but an iterative querying method, which builds a set of queries and associated weights then orders documents based on a weighted combination of their retrieval scores across the set of queries. They explore using various methods to handle query drift, including limiting the number of iterations, truncation, and interpolating with the original query. In our research we guard against query drift by using a global profile, which can be thought of as a background—or contextual—language model.

In addition to search result documents, Yin et al. [108] use external resources for query expansion, such as query logs and result snippets. Chirita et al. [19] perform query expansion by adding terms from the user's *personal information retrieval repository*, which is a collection of their text documents, emails, cached web pages, etc. A limitation of the above methods is that these resources must be accessible. Instead of requiring more data our method only uses judged documents and attempts to make better use of this existing data.

Pseudo-relevance feedback Because the Rocchio algorithm is applied on a document level it can use either explicitly judged documents or documents judged implicitly, e.g. implicit judgements may assume that a user's click indicates relevance and ignoring a document indicates lack of relevance. When the user has not judged any documents we can often achieve good results through pseudo-relevance feedback, where we assume the N highest ranked documents are relevant. A variation on this is to consider the top $M < N$ documents as relevant and then expand this set by clustering the documents and assuming the nearest neighbors of these M documents are also relevant [51].

However the usefulness of a nearest neighbors approach depends on the feature space and a poor feature space may cause the query to drift towards non-relevant documents, an effect which is amplified when diversifying over pseudo-relevant documents. In addition, a limitation of the above relevance feedback methods is that they require specific parameters, which may change depending on the user and query. When applying learning to rank we are able to optimize over multiple feature spaces and avoid having to select these parameters.

Especially in pseudo-relevance feedback, we must ensure that our reformulated query strikes an appropriate balance between the original and expanded query. From what we know about the large variance and ambiguity in user queries and their reformulations of queries [40, 81], it is naive to expect that a single interpolation weight parameter, which we will refer to as α, is appropriate across all queries and users. Lv and Zhai [61] build a system to adaptively predict the α coefficient for each query and corpus tuple by using heuristics to characterize the balance between query and feedback information. They improve retrieval performance through an adaptive interpolation coefficient, which is especially effective in the case of difficult queries and sparse domain knowledge. By modeling each query individually, our application of learning to rank is analogous to a more robust version of their method.

Reducing the feature space In our method, projecting documents into a reduced feature space before learning is an important step because it reduces sparsity, allows for feature engineering, and protects against over-fitting. Whereas Lv and Zhai [61] use a reduced feature space to select an interpolation weighting, Xu and Akella [106] use a reduced feature space to choose terms for query expansion. They first project each document's term space onto its retrieval score, its distance to relevant feedback documents, and its distance to non-relevant feedback documents; then model the probability of term relevance. Whereas these methods use a reduced feature space for parameter tuning and term selection, we use a reduced feature space for re-ranking.

Query expansion as query reformulation We may consider query expansion as a form of automatic query reformulation. Studies of search logs show that users often rewrite their queries and that in only approximately half of the cases are users satisfied with their first query [8, 9, 78]. Furthermore, users perform even more reformulations in exploratory search scenarios [95]. This indicates that we should expect a large number of reformulations in recall oriented retrieval tasks for specialized collections. The prevalence of query reformulation empirically justifies, and is the manual analog of, automatic query expansion.

Selecting documents If we are able to request relevance judgements from our users we must choose documents for judgement that best enable us to order or reorder results. Research in active learning studies how to select documents for judgement. Although we do not primarily pursue an active learning approach, in the context of iterated queries there is significant overlap between learning to rank and active learning. Whereas active learning orders documents to efficiently search through the hypothesis space [26, 90], learning to rank orders documents to maximize retrieval score.

We define an *exploitative* strategy as one that orders documents to optimize retrieval scores or user benefit, and an *explorative* strategy as one that orders documents to learn more about user preferences [72]. The active learning method of Xu and Akella [106] chooses the next document for user judgement so as to minimize the variance in error. Their method, and other active learning methods, explore—as opposed to exploit—the document space, which can detrimentally affect retrieval scores.

If the search system is expecting multiple interactions with the user, we have an incentive to highly rank documents that will provide useful information if they are judged. When we sample based on retrieval score, our learning to rank algorithm contains an embedded active learning component: we assume that the highest ranked document will be most useful for reordering. An advantage of our method, as compared to other active learning approaches, is that reordering is entirely exploitative, even if document selection is exploratory.

Modeling negative feedback Incorporating negative feedback is such a challenging problem that the standard method is to completely disregard this in-

formation and ignore documents judged non-relevant. However, Wang [98] and Wang et al. [99, 100] find that negative feedback can be especially beneficial for difficult queries and that language models are usually better at handling negative feedback than vector space models. They also report that negative feedback does not perform well with normal relevance feedback methods, and that if it is used without positive relevance information it often hurts performance.

To address the challenges of negative feedback, Xu and Akella [107] develop a relevance feedback method using the Dirichlet compound multinominal to model the overlap between positive and negative feedback documents. The positive documents are modeled as a mixture of relevant and non-relevant document models; while the negative documents are modeled as a mixture of relevant documents, non-relevant documents, and a background noise model. As opposed to many relevance feedback methods this method, as well as our method, use negative feedback documents.

When modeling relevant documents, building a single language model for all relevant documents is usually an effective approach. However, Wang [98] and Wang et al. [100] found that building a single language model for all non-relevant documents performs significantly worse than building a separate language model for each non-relevant document and then combining these models. Our method avoids the problems of modeling negative feedback by using non-relevant documents in reordering, not query expansion.

2.2 Personalized search

Whereas relevance feedback is broadly defined to cover any methods that use feedback to improve results, personalized search refers specifically to methods that use feedback from a particular user to improve results especially for that user [46]. Query expansion and search personalization are both justified by the expectation that modeling users, queries, and returned documents can improve retrieval scores. In research aimed at modeling users, Pandey and Luxenburger [70] build an updated query as a linear interpolation of the original query and terms from a search history language model. Lorenzetti and Maguitman [54] develop an incremental and topic driven method of selecting web documents in which they weight term relevance contributions according to their descriptive and discriminative power relative to user context. Luxenburger et al. [60] match user needs with previous tasks and dynamically switch from a personalized to a general retrieval model in order to alleviate problems caused by personalizing for topics that do not have enough user feedback.

Pandey and Luxenburger [70] and Luxenburger et al. [60] only apply personalization to query expansion. We go further and apply personalization to both query expansion and reordering. We move beyond controlling the amount of personalization through a simple weighting parameter that favors either the original query or the expansion terms. We build a query-specific language model and a global language model over all queries. By using the global language model when the query-specific model lacks sufficient information, we can both back-off to a more informative model and guard against over-fitting to a sparse model.

In a personalization approach using log data, Matthijs and Radlinski [64] reorder results based on the user's complete browsing history and build a personalization profile as a vector of weighted terms. Liu et al. [53] analyze a large amount of query log data to build a personalization profile. In research applied to multimedia and news documents, Hopfgartner and Jose [41] build semantic user profiles by developing a generic ontology from implicit relevance feedback based on users' interactions with the search system. All of these personalization systems rely on either user interaction or query log data, while our method is applicable with less detailed data.

Interactive personalization Although the interactivity in our research is simulated, it is similar to the ideas described by Piwowarski and Lalmas [73], in which user information needs evolve over time as a mixture of needs expressed through multiple interactions with the search system. A problem with their method is that it may fail to adequately represent the user if the assumption that they have multiple information needs is false. To avoid this problem, we assume the user has a single information need and update our model of this single need on each iteration. Balabanovic [5] infers topic interest from user actions and allows users to define topics and control importance per topic, he then learns from the adjustments users make. As above, a potential problem with this method is that it requires in-depth user interactions, whereas our method relies only on relevance feedback data.

2.3 Learning to Rank

The field of learning to rank develops methods which use machine learning algorithms to induce a re-ranking of returned documents based on document, query, and user features. Learning to rank is the closest area of research that our method borrows from and contributes to. Figure 2 shows the typical flow of a learning to rank system [52]. The learning system uses the training data, which contains relevance labels, and learns a model h for this data. Then a ranking (or classification) system uses this model to predict the relevance of unlabeled test data. If these predictions are not in the form of a ranking we can sort documents by their predicted relevance to create a ranked list of documents, which we then return to the user.

Learning to rank systems are categorized based on how they represent documents—the input space, and how they represent predictions—the output space. In the *pointwise* approach, inputs are single feature vectors and outputs are the relevance of each document. In the *pairwise* approach, inputs are pairs of document feature vectors that encode document preferences, and outputs are pairwise preferences between all documents. Finally, in the *listwise* approach the input is the entire group of returned documents for a query, and the output is either the relevance of each document or a ranked list of these documents.

We learn to rank using the *pairwise* approach, for which there are a number of popular and effective algorithms [12–14, 30, 44, 91]. The pairwise approach

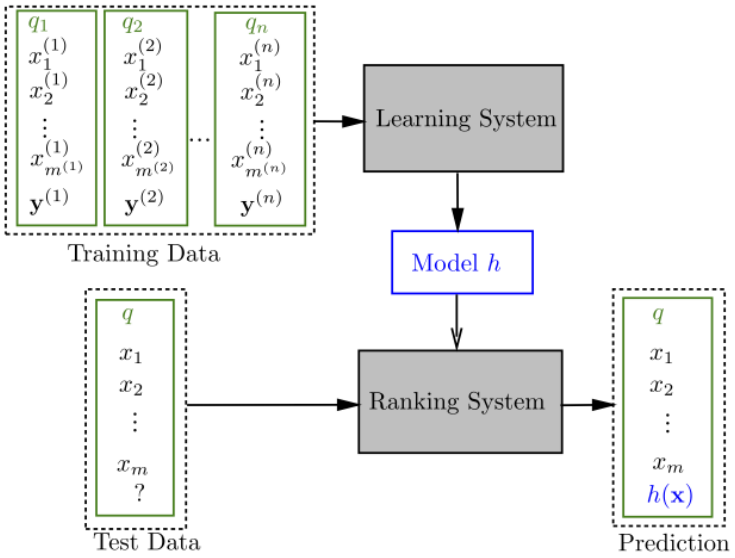


Fig. 2. Learning to rank framework, image taken from Liu [52].

reduces the ranking problem to that of minimizing the number of misclassified document pairs, or *preference pairs*. We apply Ranking SVM [44] to solve this classification problem. Ranking SVM uses support vector machines [93] to find a weighting that minimizes the number of incorrectly ordered pairs of labeled instances, and then uses this weighting to predict the relevance of unlabeled instances. It is well known that SVMs can have difficulty with unbalanced datasets [3], however Ranking SVM [45] is not sensitive to class imbalance because forming pairs of positive and negative documents balances the data [94].

The Ranking SVM system is based on SVM^{light} [43] but was improved significantly in SVM^{rank} [45] and RankSVM [17]. Our research involves a novel application of Ranking SVM which uses the SVM^{rank}[7] and SGD-SVM[8] [84] implementations. The SGD-SVM method improves performance by implicitly sampling from the set of pairwise document preferences instead of explicitly constructing it. Because SGD-SVM is not dependent on the size of the set of preference pairs it can reduce the training time by orders of magnitude (when compared to methods that create the full set of preference pairs) without decreasing ranking performance.

Learning to rank for interactive and specialized collection search Verberne et al. [94, 96] apply a suite of learning to rank algorithms to question answering data using a specially developed set of linguistically motivated features.

[7] `http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html`
[8] `https://code.google.com/p/sofia-ml/`

As expected, they found that if the data is balanced, and hyper-parameters are properly tuned, all the learning to rank methods tested outperform the original retrieval ordering, which assigned relevance based upon term-frequency inverse-document-frequency. In another application of learning to rank, D'hondt et al. [27] use syntax based dependency triplets as features for re-ranking prior-art in the context of patent search (the CLEF-IP track). Both this research and our method apply learning to rank to document based features. While the research in [27, 94, 96] is limited to single queries, we evaluate our learning to rank method in the broader context of search interactions which involve multiple queries per information need.

In recent work, Brandt et al. [10] use a dynamic ranking system which changes in response to user interactions. Their method iteratively builds a ranked list by adding the document with the highest expected gain in utility—where any standard retrieval metric can be used to measure utility. In the system Brandt et al. [10] construct, users interact with results through a "dynamic ranking tree," which contains multiple generated ranked lists. Although their work is quite different from ours on this presentational level, the concept of learning from user interaction to reorder documents—we make use of relevance judgements—is analogous to the motivation behind our method.

Diversity and novelty Although learning to rank is a powerful method for exploiting relevance feedback, problems can arise when it is applied to ambiguous queries, poorly defined queries, or queries that do not represent the user's true information need. These queries often highly rank non-relevant documents, and if learning to rank is subsequently applied this can worsen the problem by focusing on a non-relevant subtopic of one of these non-relevant documents. Techniques in search result diversity and novelty address these problems by reordering results to ensure that highly ranked results cover a variety of aspects or topics, which may be intended by the query [1, 16, 20, 55].

A problem with standard diversification algorithms is that they require given or learned subtopics for each query and document. Although we do not explicitly apply search diversification algorithms in our research, the learning to rank approach we use has an intrinsically diversifying component and obviates the need for subtopics. When using the global profile to reorder based on the features in documents for queries other than the current query, we are "diversifying" over the features of topics we expect the user is interested in but are not directly related to the query at hand.

Our use of implicit diversification, instead of an explicit result diversification algorithm (e.g. [1, 20]), is justified by recent work that contrasts the suitability of relevance and diversity metrics as objective functions for learning a diverse ranking. Santos et al. [82] find that diversity oriented metrics cannot outperform relevance-oriented metrics (e.g. relevance-oriented mean average precision, MAP, compared to diversity-oriented intent aware mean average precision, MAP-IA) and also generally perform worse than diversity metrics that have been modified to not penalize redundancy.

Our method is built to address a recall oriented task, which relates to research that applies search diversification to informational queries. In an informational query the user's information needs are expected to require more than one page of search results, precisely the scenario we address. In research aimed at informational queries, Welch et al. [102] develop the Diversify-IQ algorithm, which returns an ordering that trades off the desire for multiple relevant documents, probabilistic user interest information about query subtopics, and uncertainty in classifying documents into these subtopics.

Related work by Santos et al. [83] diversifies results by learning the appropriateness of different retrieval models for each subtopic (which they refer to as an *aspect*) underlying the query. As with most diversification algorithms, this limits us to datasets with known or easily discoverable subtopics. Our method does not require that we know any subtopics. Without explicitly modeling subtopics, we use judged documents to gather subtopic information about our query, which we then use for reordering.

Learning to rank as semi-supervised learning We can view learning to rank as a semi-supervised learning task in which judged documents have labels and other documents are unlabeled. Our goal is to classify the unlabeled documents using the information from the labeled documents (this overlooks the ordering produced by Ranking SVM). When a portion of our data lacks labels we can apply methods from semi-supervised learning [18, 48]. Nigam et al. [67] apply semi-supervised learning to classify text documents using expectation maximization and deterministic annealing, which reduces problems with local maxima. Their method involves multiple classes, which differs from the binary pairwise loss minimization we use in our application of learning to rank.

2.4 Recall oriented TREC tracks

The annual TREC Legal Track began in 2006. It focuses on developing recall oriented retrieval methods for legal search, especially in the context of e-discovery. The TREC 2009 and 2010 Legal Tracks use a dataset consisting of emails from the Enron Corporation and the retrieval tasks relate to the court cases surrounding Enron's accounting practices.[9] The TREC 2010 Legal Track contains two tasks: an interactive task in which a *topic authority* is available to judge documents, and a learning task with a fixed *seed* set of judged documents. Document judgements are made by specially assigned assessors and, as in any information retrieval dataset, the judgements are not perfect. Gordon V. Cormack, an organizer of the TREC 2010 and 2011 Legal Tracks, stated:

> A consequence of [this] is that teams should assume a small level of unreliability in the relevance determinations they are given.[10]

[9] For additional background on the Enron court cases refer to, `https://secure.wikimedia.org/wikipedia/en/wiki/Enron_scandal`.

[10] Correspondence on TREC 2011 Legal mailing list, `https://mailman.umiacs.umd.edu/pipermail/trec-legal/2011/000632.html`.

In the TREC Legal Track evaluation there is even an *appeals* process, whereby participants can challenge a relevance judgement that they think is incorrect. We will overlook these details and focus on the TREC Legal *learning task*, where the goal is to order all documents for a *discovery request* from most to least relevant.

Participants' results in TREC Legal 2009 were substantially better than random. Participants used training examples from previous years' interactive tasks to attain, in non-interactive tasks, at least the same scores as in those previous years' tasks [36]. In TREC Legal 2010 the results again improved over those in previous years and the learning task, which our results are directly comparable to, was first introduced [23]. As described by Renders [77], one of the best performing methods in the TREC Legal 2010 learning task used an up to trigram language model in combination with clustering. Our method complements this by additionally using retrieval scores as features and learning incrementally.

In other approaches, Garron and Kontostathis [31] apply latent semantic indexing to reduce dimensions, and Papadopoulus et al. [71] use kernel density estimation and logistic regression to model the training data. Tomlinson [89] use relevance feedback based upon the seed documents, and Culotta et al. [24] use a Naive Bayes classifier to model the seed documents. The Naive Bayes classifier is known to have problems with unbalanced datasets [94, 96]. We avoid these problems by using Ranking SVM.

These methods are similar to those used in the TREC-CHEM track, which involves a prior art search task and is similarly recall oriented [57]. Given this overlap, we expect that applying our research to patent search will be a fruitful avenue for future work.

3 Method

We build a search system that learns from user feedback to re-rank documents and align them with user preferences. Our system performs online learning as more judged documents are collected and learns from different features of these judged documents. The main components of our search system are:

1. a retrieval function that ranks documents,
2. an expansion function that builds updated queries from judged documents,
3. a learning function that learns a document ranking using judged documents.

We formulate our system abstractly so that these various functions can be updated as methods and researcher needs change.

The search process begins with standard retrieval, then as the user browses and judges results, we iteratively add judged documents to a query-specific and global profile. We next use these documents to expand our original query and retrieve updated results, then to reorder these results through learning to rank using features extracted from the judged documents. Below we first describe the retrieval method and then the different sampling methods we use to simulate the process of selecting judged documents. Next we describe our query expansion and learning to rank methods, along with the feature spaces we use to represent documents for learning to rank.

3.1 Retrieval method

We assume a corpus $\mathcal{C}$, which contains documents $\mathbf{d} \in \mathcal{C}$; a set of queries $\mathbf{q} \in \mathcal{Q}$; and a set of judged documents for each query $\{\langle \mathbf{d}, j \rangle | \mathbf{d} \in D_{\mathbf{q}}\}$, where $j \in \{1, -1\}$ indicates that the document $\mathbf{d}$ is respectively relevant or non-relevant. We index and retrieve documents with the Lemur (Indri 5.0)[11] search engine using language modeling. We run queries under the default retrieval model [74, 92], which uses Dirichlet smoothing with $\mu = 2500$. To formalize this, let $\mathcal{R}$ be a function which assigns retrieval scores s_i to the documents $\mathbf{d}_i$ of a corpus:

$$\mathcal{R}(\mathbf{q}, \mathcal{C}) = \langle s_1, \ldots, s_n \rangle, \tag{1}$$

for a query $\mathbf{q}$ and corpus $\mathcal{C}$ containing n documents, $|\mathcal{C}| = n$. This induces an ordered set of documents:

$$\mathcal{R}_D(\mathbf{q}, \mathcal{C}) = \langle \mathbf{d}_1, \ldots, \mathbf{d}_n \rangle, \tag{2}$$

where for $\mathbf{d}_i$ and $\mathbf{d}_j$, $i < j$ iff $s_i > s_j$ (ties are broken arbitrarily).

[11] `http://www.lemurproject.org`

3.2 Query expansion

We learn from judged documents on two levels by using query expansion in concert with learning to rank. In query expansion we use the judged documents, $D_{\mathbf{q}}$, to build a modified query:

$$\tilde{\mathbf{q}} = \alpha \mathbf{q}_0 + (1 - \alpha) \sum_{\mathbf{d}_j \in D_{\mathbf{q}}} \mathbf{d}_j, \tag{3}$$

where $\mathbf{q}_0$ is the original query and α is an interpolation parameter. We retrieve documents with the expanded query $\tilde{\mathbf{q}}$, which provides us with a retrieval score per document. We will use these retrieval scores as a feature in learning to rank and for our baseline run before reordering.

To choose terms for query expansion we require a method of representing corpus contents that indicates which terms are more relevant. To represent the terms in a corpus of one or more documents, we define a function $\mathcal{L}$ that returns a set of ordered n-grams:

$$\mathcal{L}(\mathcal{C}) = \langle t_1, \ldots, t_m \rangle, \tag{4}$$

where t_i is more relevant to $\mathcal{C}$ than t_j iff $i < j$. In our experiments, $\mathcal{L}$ uses terms (1-grams) and defines relevance to a corpus as term frequency (count) within that corpus.

Using the document profiles we build language models for their relevant and non-relevant documents and then use these language models to expand the query, as described in equation 3. This is similar to the standard equation used in the Rocchio algorithm [62]:

$$\tilde{\boldsymbol{q}} = \alpha \boldsymbol{q}_0 + \beta \frac{1}{|D_r|} \sum_{\boldsymbol{d}_j \in D_r} \boldsymbol{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\boldsymbol{d}_j \in D_{nr}} \boldsymbol{d}_j, \tag{5}$$

however we set $\gamma = 0$ and expand without using non-relevant documents, although we make use of them later in the learning to rank stage. To describe query expansion, we let $\circ$ denote string concatenation, and transform the baseline query $\mathbf{q}_0$ into $\mathbf{q}_D \Leftarrow \mathbf{q}_0 \circ t_1 \circ \cdots \circ t_l$ by expanding it with the l most frequent terms in a language model for the document set D, $\{t_i \in \mathcal{L}(D) | i \leq l\}$.

Indri syntax for query expansion In the Indri syntax used by Lemur we write query expansion as in Figure 3. We follow the standard method and interpolate the original query with weight α and the expansion terms with weight $1 - \alpha$. Indri converts the above structured query into a belief network where operators are nodes with a belief b_i and, for weight operators, a weight w_i. The `weight` operator is defined with respect to its n child nodes—which in the above query are the two `combine` operators—as:

$$\texttt{weight} = \prod_{i=1}^{n} b_i^{(w_i / W)},$$

Generic notation:

```
weight(
  α combine( [original query] )
  (1 − α) combine( [positive terms] )
)
```

Our notation:

```
weight(
  α combine( q_0 )
  (1 − α) combine( t_1 ∘ ··· ∘ t_5 | t_i ∈ L(P_q) )
)
```

Fig. 3. Indri syntax for query expansion in generic notation (top) and our notation (bottom).

where W is the sum over all weights. The `combine` operator is defined as:

$$\texttt{combine} = \prod_{i=1}^{n} b_i^{(1/n)},$$

where n is again the number of child nodes, and these nodes are now the terms passed as arguments to `combine` [87].

Judged document profiles We simulate building user profiles by sampling judged documents as the user interacts with our search system. We experiment with three sampling strategies:

1. *exploitative feedback*: sample descending from the highest ranked document,
2. *random seed*: randomly sample from the seed set of judged documents,
3. *random complete*: randomly sample from all judged documents.

Strategy 1 simulates an interactive retrieval process and is closest to what we would expect in a real-life search system, because users are presented with documents descending from the highest ranked. In Strategy 2, the seed documents are a given set of informative judged documents from our corpus. Strategy 2 represents a scenario in which we have identified informative documents that we would like to be judged. Strategies 1 and 2 are directly comparable to the results in TREC Legal 2010.

Strategy 3 is an oracle run and not comparable to the TREC Legal 2010 results because it has access to judged documents outside of the seed set. Strategy 3 may underperform because only the population of documents we are sampling from is larger, and not the number of samples being taken. More importantly, the seed documents are chosen to be informative about the collection, therefore when we sample from the seed set we expect to encounter documents that are more useful for reordering. When sampling from a larger collection of documents we decrease the probability that we will encounter an informative document.

Strategy 1, exploitative sampling, will bias the collection of judged documents towards those ranked higher. We expect this will have greater negative than

positive effects, because it gives us an incomplete picture of our feature space. When sampling randomly, we are more likely to select a relevant low ranked document, and learning to rank with this document could increase the rank of other relevant but incorrectly low ranked documents, which may significantly improve scores.[12]

Algorithm 1 precisely describes the above sampling processes, where '\' is the set difference and `randomItemFrom` is a (pseudo-)random sampling function which takes a set as its argument and returns one item from this set. The required

Algorithm 1 Sampling judged documents.

Require: $\mathcal{C}, D_{\mathbf{q}}, D$
  $P_{\mathbf{q}} \leftarrow \{\}$
  $i \leftarrow 0$
  while $|P| <$ desired size of profile **do**
    if exploitative (1) **then**
      $P'_{\mathbf{q}} \leftarrow D_{\mathbf{q}} \cap \{\mathbf{d}_i \in D\}$
      $i \leftarrow i + 1$
    else if random seeds (2) **then**
      $P'_{\mathbf{q}} \leftarrow$ `randomItemFrom`$(D_{\mathbf{q}} \setminus P_{\mathbf{q}})$
    else if random all (3) **then**
      $P'_{\mathbf{q}} \leftarrow$ `randomItemFrom`$(\mathcal{C} \setminus P_{\mathbf{q}})$
    end if
    $P_{\mathbf{q}} \leftarrow P_{\mathbf{q}} \cup P'_{\mathbf{q}}$
  end while
  return $P_{\mathbf{q}}$

data structures are a corpus of documents $\mathcal{C}$, for when randomly sampling from all documents; the set of judged seed documents $D_{\mathbf{q}}$, for when sampling from the seed set; and the ordered set of returned documents D, for when sampling descending from the highest ranked document. The algorithm adds documents to the query-specific profile $P_{\mathbf{q}}$ until it is of whatever size the system desires, and then returns $P_{\mathbf{q}}$. In our implementation we build a relevant and non-relevant document profile, both of which contain the same number of documents, or as close to this as possible given the total number of judged documents. Algorithm 1 omits these trivial details.

Query expansion using judged document profiles We expect that by focusing on the current context of the user's information need, query expansion based on documents specifically judged relevant to a query will improve results beyond a general—and not query-specific—profile of relevant documents for the

[12] With exploitative sampling it is possible to select documents that will have a low rank in future iterations. A highly ranked document may be sampled in one iteration and then, after query expansion, that document's rank may decrease significantly, but it will remain an instance used in learning.

user. We build a profile $P_{\mathbf{q}}$ for query $\mathbf{q}$ by repeatedly sampling, as described in Algorithm 1, and then we use this profile to build a language model, $\mathcal{L}(P_{\mathbf{q}})$. Given this language model we can expand our query specifically for $P_{\mathbf{q}}$:

$$\mathbf{q}_{P_{\mathbf{q}}} \Leftarrow \mathbf{q}_0 \circ t_1 \circ \cdots \circ t_l \tag{6}$$

where we take the l most frequent terms, $\{t_i \in \mathcal{L}(P_{\mathbf{q}}) | i \leq l\}$. Although term frequency may be a sub-optimal way to select terms, it has been shown that word frequency in English text obeys a power law distribution, known as Zipf's law [86, 113], and it is therefore a reasonable method for selecting meaningful words from a text corpus [29, 62, 103].

If the query-specific profile is too narrow our search system may benefit from a broader profile of documents, which represents the user's more general information needs. We build a global document profile P_* by taking one document from each query-specific profile. We then focus the global profile on a query by combining it with the query-specific profile: $P_{\mathbf{q}*} \leftarrow P_* \cup P_{\mathbf{q}}$. Using the global profile for query $\mathbf{q}$ we apply query expansion as above:

$$\mathbf{q}_{P_{\mathbf{q}*}} \Leftarrow \mathbf{q}_0 \circ t_1 \circ \cdots \circ t_l,$$

where we take the l most frequent terms, $\{t_i \in \mathcal{L}(P_{\mathbf{q}*}) | i \leq l\}$, to build the more diversified query $\mathbf{q}_{P_{\mathbf{q}*}}$. The global profile attempts to capture document characteristics which are relevant for any production request.

Building query expansion sets In our retrieval score feature spaces we create a representation of our corpus relative to its judged documents by compiling retrieval scores for queries expanded with a varying number of terms from these judged documents, or for different sets of judged documents. Using successivezly modified queries we build tuples of data:

$$\{\mathbf{q}_{P_{\mathbf{q}}}, P_{\mathbf{q}}, \langle s_1, \ldots, s_n \rangle\},$$

where each tuple respectively includes the modified query $\mathbf{q}_{P_{\mathbf{q}}}$; the judged documents $P_{\mathbf{q}}$; and the retrieval scores per document $\langle s_1, \ldots, s_n \rangle$, where we have retrieved n documents from our corpus. We then use these retrieval scores to reorder our corpus, pushing documents similar to relevant documents higher in ranking, and the contrary for non-relevant documents. Our method learns a reordering function:

$$\begin{aligned}
\mathfrak{R} : \langle & \\
& \{\mathbf{q}_0, P_{\mathbf{q}}, \langle s_1, \ \ldots, s_n \rangle\}, \\
& \{\mathbf{q}_{P'_{\mathbf{q}}}, P'_{\mathbf{q}}, \langle s_1, \ldots, s_n \rangle\}, \\
& \{\mathbf{q}_{P''_{\mathbf{q}}}, P''_{\mathbf{q}}, \ \langle s_1, \ldots, s_n \rangle\}, \\
& \ldots \\
& \rangle \rightarrow \langle \mathbf{d}_1, \ldots, \mathbf{d}_n \rangle,
\end{aligned}$$

which uses the query expansion tuples to produce a reordered set of documents, $\langle \mathbf{d}_1, \ldots, \mathbf{d}_n \rangle$, as described below in Sections 3.3 to 3.5.

3.3 Learning to rank

We learn to rank our corpus by using relevant documents as positive instances and non-relevant documents as negative instances. We will reorder the top m documents returned by our retrieval method when using the expanded query. As noted in relevance feedback research [112], query expansion can lead to detrimental query drift in which judged documents are relevant to a specific topic that is irrelevant to the user's information need, and the expansion terms "drift" towards this irrelevant topic. By using learning to rank we can potentially correct this drift.

Learning method To learn a reordering of our retrieved documents we begin by projecting the judged documents into different feature spaces to create a feature vector $\mathbf{x}_i = \Phi(\mathbf{d}_i, \mathbf{q})$, where Φ extracts features from document $\mathbf{d}_i$ and query $\mathbf{q}$, this is described below in Section 3.5. We use document judgements to build a set of preference pairs $\mathcal{P}$ such that $(i, j) \in \mathcal{P}$ iff document $\mathbf{d}_i$ is preferred to $\mathbf{d}_j$. We then minimize the objective function:

$$\frac{1}{2}||\mathbf{w}||^2 + C \sum_{(i,j)\in\mathcal{P}} \ell(\mathbf{w}^\top \mathbf{x}_i - \mathbf{w}^\top \mathbf{x}_j), \tag{7}$$

where C is a parameter that allows trading-off margin size against training error and ℓ is an appropriate loss function, in our case it is the squared hinge loss: $\ell(t) = \max(0, 1 - t)^2$ [17, 34]. Minimizing Equation 7 gives us $\mathbf{w}$, the weighting vector over the feature space which minimizes the number of incorrectly ordered pairs of judged documents.[13]

We then project the top m documents from the corpus—as ranked by query $\mathbf{q}$—into the same feature space and use $\mathbf{w}$ to classify these documents:

$$\mathbf{y} = \mathbf{w}^\top \mathbf{X},$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]$ are feature vectors for documents $[\mathbf{d}_1, \ldots, \mathbf{d}_m]$, and each value $y_i \in \mathbf{y}$ is the predicted relevance of document $\mathbf{d}_i$. We sort the documents by their predicted relevance, as given by $\mathbf{y}$, and append the not-reordered documents ranked greater than m, to produce a reordering of the entire corpus. We describe the reordering function by:

$$\mathcal{S}_\mathcal{P}(D, \mathbf{q}) = \mathbf{w}^\top \mathbf{\Phi}(D_m, \mathbf{q}) \cup \mathcal{R}(\mathbf{q}, D_{\overline{m}}), \tag{8}$$

where $D_m = \{\mathbf{d}_i | \mathcal{R}_D(\mathbf{q}, D) \wedge i \leq m\}$, the top m documents in D ordered by retrieval with $\mathbf{q}$; $D_{\overline{m}} = D \setminus D_m$; and the subscript $\mathcal{P}$ indicates we use the set of preferences pairs $\mathcal{P}$ to minimize Equation 7. $\mathbf{\Phi}$ is the feature extractor which returns $\Phi(\mathbf{d}_i, \mathbf{q})$ as a column vector for every $\mathbf{d}_i \in D$.

[13] We use SGD-SVM as implemented in sofia-ml, `https://code.google.com/p/sofia-ml/`, which solves equation 7 with stochastic gradient descent.

Defining preference pairs Where P is an arbitrary judged document profile, we define the preference pairs $\mathcal{P}$ as:

$$\{(i,j) \in \mathcal{P} | \langle \mathbf{d}_i, 1 \rangle \in P \wedge \langle \mathbf{d}_j, -1 \rangle \in P\}, \tag{9}$$

stating that (i,j) is in $\mathcal{P}$ iff document $\mathbf{d}_i$ is relevant and $\mathbf{d}_j$ is non-relevant. We will use the query-specific profile $P_{\mathbf{q}}$ and the global profile $P_{\mathbf{q}*}$ to respectively form the preference pairs $\mathcal{P}_{\mathbf{q}}$ and $\mathcal{P}_{\mathbf{q}*}$. We can now learn to rank with these preference pairs and rank with the reordering functions $\mathcal{S}_{\mathcal{P}_{\mathbf{q}}}$ and $\mathcal{S}_{\mathcal{P}_{\mathbf{q}*}}$.

To keep track of our iterative profiles and reordering functions we introduce a counter $P_{\mathbf{q}}^{(t)}$ where t is the number of times we have added documents to $P_{\mathbf{q}}$, and equivalently the number iterations.[14] Each iteration provides us with more judged documents, which we use to produce a more informed ranking. These judged documents form sets of preferences pairs $\mathcal{P}_{\mathbf{q}}^{(t)}$, which we then use for learning to rank on the tth iteration through the reordering function $\mathcal{S}_{\mathcal{P}_{\mathbf{q}}^{(t)}}$.

3.4 Iterative search algorithm

With retrieval, sampling, query expansion, and learning to rank, we have all we need for the algorithm used in our iterative search system. Figure 4 describes the processes used in our search system. It is equivalent to Figure 1, except we have replaced text descriptions with the notation defined in this section.

We present our retrieval system as Algorithm 2, where the function `sample` is a reference to Algorithm 1. Algorithm 2 requires a retrieval function $\mathcal{R}_D$, which returns a document ordering; a corpus $\mathcal{C}$; the set of queries $\mathcal{Q}$, where we refer to the unexpanded queries as $\mathbf{q}_0 \in \mathcal{Q}$; and the seed documents for each query, $\{D_{\mathbf{q}} | \mathbf{q} \in \mathcal{Q}\}$.

For each query, we run the first retrieval with the unexpanded query $\mathbf{q}_0$, on line 4, then loop until the user is satisfied. In each iteration of the loop we update the user profile, on line 6, by sampling one or more judged documents. We use the user profile in another retrieval run with an expanded query, on line 7, and then again in learning to rank, on line 8. Once the user is satisfied we add the first judged document in the query-specific profile ($P_{\mathbf{q},0}$) to the global profile, on line 10, and continue with the next query.

To build a global profile for each query we take the union of the global profile and the query-specific profile. This ensures that there are more, or at the least an equal number of, judged documents for the query we are evaluating as there are for any other queries in the global profile. We note that the pairwise loss function can be significantly affected by queries with a large number of judged documents pairs [52]. This is an advantage in our retrieval setting because we want documents judged for the current query to have a greater influence on reordering than documents for other queries.

[14] We are overloading t, which is also used to represent a term. However there should be no confusion because t_i, as in term i, will *always* have an index and t, as in the tth iteration, will *never* have an index.

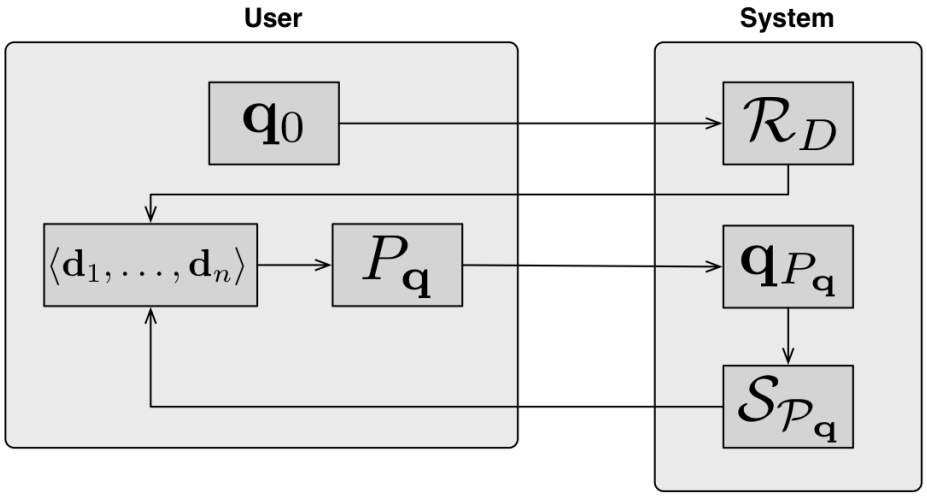


Fig. 4. Diagram of our method. $\mathbf{q}_0$ is the original query, it is passed to the retrieval function $\mathcal{R}_D$, which returns documents $\langle \mathbf{d}_1, \ldots, \mathbf{d}_n \rangle$. The user judges a subset of these documents, $P_\mathbf{q}$, which the system uses to build the modified query $\mathbf{q}_{P_\mathbf{q}}$. Then preference pairs are built from the judged documents and used to reorder the results—which have been retrieved using the expanded query—with the learning to rank function $\mathcal{S}_{\mathcal{P}_\mathbf{q}}$. The newly ordered results are returned to the user and the process continues until her information needs are satisfied.

Algorithm 2 Query expansion and learning to rank with a query-specific profile, where the function `sample` is a reference to Algorithm 1. We define the condition "information need unsatisfied" as false after a maximum number of iterations.

Require: $\mathcal{R}_D, \mathcal{C}, \mathcal{Q}, \{D_\mathbf{q} | \mathbf{q} \in \mathcal{Q}\}$
for $\mathbf{q} \in \mathcal{Q}$ **do**
  $P_* \leftarrow \{\}$
  $P_\mathbf{q} \leftarrow \{\}$
  $D \leftarrow \mathcal{R}_\mathcal{D}(\mathcal{C}, \mathbf{q}_0)$
  while information need unsatisfied **do**
    $P_\mathbf{q} \leftarrow P_\mathbf{q} \cup \texttt{sample}(\mathcal{C}, D_\mathbf{q}, D)$
    $D \leftarrow \mathcal{R}_\mathcal{D}(\mathcal{C}, \mathbf{q}_{P_\mathbf{q}})$
    $D' \leftarrow \mathcal{S}_\mathcal{P}(D, \mathbf{q})$
  end while
  $P_* \leftarrow P_* \cup P_{\mathbf{q},0}$
end for

Although this is not depicted in Algorithm 2, if we assume P_* has been populated by running retrieval for all other queries, we can replace line 6 by $P_{\mathbf{q}*} \leftarrow P_* \cup P_{\mathbf{q}} \cup \texttt{sample}(\mathcal{C}, D_{\mathbf{q}}, D)$ to depict building a global profile for query $\mathbf{q}$. We would then use the global profile for query expansion and learning to rank. In a real system the while-loop condition, "information need unsatisfied," is false when the user ceases interacting with the system. For our simulation, and to better compare across queries, after a maximum number of iterations we assume the information need is satisfied and break the while-loop.

3.5 Feature spaces

As shown in Equation 7, we learn a weight vector from the document feature vectors, not the documents themselves. Therefore extracting representative features from documents is an essential part of learning to rank. We experiment with three different feature spaces: two using variations on retrieval score features and one using term frequency. In the two retrieval score feature spaces we calculate each document's features as that document's retrieval scores using a *cumulative* and a *constant* method, which are described below.

In the term frequency feature space we experiment with different size term spaces as well as hybrid methods that combine term frequency features with retrieval score features. By adding retrieval scores as a feature the hybrid methods put greater weight on the original ordering and thereby reduce the effect of term frequency features on the order. If reordering with term frequency features is detrimental to scores the hybrid method may improve performance by reducing their influence, however it may also dampen the beneficial influence of term frequency features.

Cumulative retrieval score features In the cumulative method, we accumulate features from the retrieval scores for queries expanded with different judged document profiles as documents are added to these profiles. The cumulative method chooses the document profiles which produce the best ordering. After the tth expansion iteration, we have collected retrieval scores for the original query $\mathbf{q}_0$ and for queries $\{\mathbf{q}_i | i \leq t\}$, the queries after expansion using up to and including the tth iteration's set of judged documents. The cumulative method takes tuples of these queries, the document profile, and retrieval scores; then returns the scores for the top m documents that will be reordered:

$$\mathbf{\Phi}\left(\begin{bmatrix} \{\mathbf{q}_0, \emptyset, \langle s_1^{(0)}, \ldots, s_n^{(0)} \rangle\} \\ \{\mathbf{q}_1, P_{\mathbf{q}}^{(1)}, \langle s_1^{(1)}, \ldots, s_n^{(1)} \rangle\} \\ \vdots \\ \{\mathbf{q}_t, P_{\mathbf{q}}^{(t)}, \langle s_1^{(t)}, \ldots, s_n^{(t)} \rangle\} \end{bmatrix}\right) = \begin{bmatrix} s_1^{(0)}, \ldots, s_m^{(0)} \\ s_1^{(1)}, \ldots, s_m^{(1)} \\ \vdots \\ s_1^{(r)}, \ldots, s_m^{(r)} \end{bmatrix}, \tag{10}$$

where $P_{\mathbf{q}}^{(i)}$ is the set of judged documents after i iterations, and before the first query expansion iteration no documents have yet been judged, implying that

$P_{\mathbf{q}}^{(0)} = \emptyset$. The number of rows in the right hand side output matrix of Equation 3.5 equals the number of features, which is less than or equal to the number of rows in the input matrix: $r \leq t$. The features for document $\mathbf{d}_i$ are the retrieval scores in the ith column of the output matrix.

We believe the cumulative method is more applicable to exploratory search than the constant method (explained in the proceeding section) because it constructs a single feature per document set and therefore puts less emphasis on, and trust in, each document judgement. A practical advantage of the cumulative method is that it uses fewer computational resources. This is possible because it relies on features from past retrieval runs and does not perform any additional retrieval runs when generating features.

A disadvantage of the cumulative method is that the number of features differs per iteration. At each new iteration we have the retrieval results for a new set of judged documents, and an additional feature to use for reordering in learning to rank. To reduce the complexity of our model, we experiment with limiting the number of features by setting a maximum number of iterations during which we add features, i.e. $r = c$ for some constant c. After we stop adding features we continue to learn by adding instances as we sample more judged documents.

While adding features, the changing number of features is a compounding factor in our analysis of the effectiveness of using additional profile documents. A higher evaluation score in later iterations—using a profile with a greater number of judged documents—may be because the documents in the profile better represent the user's information need, or merely because there are a greater number of features, which is irrelevant and uninformative with respect to our research questions. A greater number of features could also negatively affect evaluation scores if features in the new feature-dimension are less correlated with document judgements than features in the existing feature-dimensions. It is most likely that a change in evaluation score will reflect both changes in the feature space and in the number of instances.

As an example, suppose there are three documents, i.e. $n = 3$, the retrieval scores before expansion are $\langle 1, 2, 3 \rangle$, and on the 1st iteration the retrieval scores are $\langle 1, 1, 3 \rangle$. Furthermore, suppose that after the first iteration the query profile is $P_{\mathbf{q}} = \{\mathbf{d}\}$ and the expansion term is t_1. Assume that we are only collecting features up to and including the first iteration, i.e. $r = 1$, and we are reordering the top two documents, i.e. $m = 2$. Our input to the cumulative feature extractor $\mathbf{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0, \emptyset, \langle 1, 2, 3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 1, 1, 3 \rangle\} \\ \vdots \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 1, 2 \\ 1, 1 \end{bmatrix}.$$

The features are the retrieval scores without expansion and from the first iteration. The features are $[1,1]^\top$ for the first document and $[2,1]^\top$ for the second document.

Constant retrieval score features In the constant method, we build features using the current judged document profile in multiple retrieval runs with different queries constructed by varying the number of expansion terms. The constant method chooses the number of expansion terms which produces the best ordering. On the ith iteration the constant feature extractor takes the document profile $P_{\mathbf{q}}^{(i)}$, r different expanded queries $\{\mathbf{q}_0 \circ t_1 \circ \cdots \circ t_{l_j} | j \leq r\}$, and the retrieval scores when using these expanded queries:

$$\mathbf{\Phi}\left(\begin{bmatrix} \{\mathbf{q}_0 \circ t_1 \circ \cdots \circ t_{l_1}, P_{\mathbf{q}}^{(i)}, \langle s_1^{(1)}, \ldots, s_n^{(1)} \rangle\} \\ \vdots \\ \{\mathbf{q}_0 \circ t_1 \circ \cdots \circ t_{l_r}, P_{\mathbf{q}}^{(i)}, \langle s_1^{(r)}, \ldots, s_n^{(r)} \rangle\} \end{bmatrix}\right) = \begin{bmatrix} s_1^{(1)}, \ldots, s_m^{(1)} \\ \vdots \\ s_1^{(r)}, \ldots, s_m^{(r)} \end{bmatrix}, \quad (11)$$

where $\{l_j | j \leq r\}$ indicates the number of terms used in expansion, e.g. if $l_j = 10$ we expand the query with 10 terms and $\langle s_1^{(j)}, \ldots, s_n^{(j)} \rangle$ are the retrieval scores when using this expanded query. The constant method forms all of its features from the same judged document set, and thus the specific set of judged documents has a greater influence on the feature space and reordering, making it a more exploitative strategy than the cumulative method.

Furthermore, as opposed to the cumulative method, the constant method has the practical disadvantage that it requires a retrieval run for each feature. For example, to test re-ranking with r features each iteration requires r retrieval runs: one to generate each additional feature for each expanded query. However, by using the same number of features per iteration, the constant method removes the confounding effects caused by changing the number of features.

As an example, suppose there are three documents, i.e. $n = 3$, the query profile is $P_{\mathbf{q}} = \{\mathbf{d}\}$, and the first two expansion terms are $\{t_1, t_2\}$. Furthermore, suppose that the retrieval scores when expanding with t_1 are $\langle 2, 2, 3 \rangle$ and that when expanding with t_1 and t_2 they are $\langle 1, 2, 3 \rangle$. Assume that we are collecting features for expansion with the number of expansions terms $l_1 = 1$ and $l_2 = 2$, i.e. $r = 2$; and that we are reordering the top two documents, i.e. $m = 2$. Our input to the constant feature extractor $\mathbf{\Phi}$ will be:

$$\begin{bmatrix} \{\mathbf{q}_0 \circ t_1, \{\mathbf{d}\}, \langle 2, 2, 3 \rangle\} \\ \{\mathbf{q}_0 \circ t_1 \circ t_2, \{\mathbf{d}\}, \langle 1, 2, 3 \rangle\} \end{bmatrix},$$

and the output features will be:

$$\begin{bmatrix} 2, 2 \\ 1, 2 \end{bmatrix}.$$

The features are the retrieval scores for the query $\mathbf{q}_0 \circ t_1$ and for the query $\mathbf{q}_0 \circ t_1 \circ t_2$. The features are $[2,1]^\top$ for the first document and $[2,2]^\top$ for the second document.

Term frequency features To build a term frequency feature space, we form vectors consisting of the top l terms in the global and query-specific profiles. To provide better coverage of terms, and increase the likelihood that all instances have at least one feature, we augment this vector with the top u terms from the set of documents to be reordered. Recalling the language modeling function $\mathcal{L}$ introduced in equation 4, we build term based features for every query as:

$$\mathbf{f_q} = \{t_i | (t_i \in \mathcal{L}(P_\mathbf{q}) \wedge i \leq l) \vee (t_i \in \mathcal{L}(D_m) \wedge i \leq u)\}, \tag{12}$$

where $P_\mathbf{q}$ is the profile for query $\mathbf{q}$ and D_m equals the first m documents returned by $\mathcal{R}_D(\mathcal{C}, \mathbf{q}_{P_\mathbf{q}})$, a retrieval run with the expanded query $\mathbf{q}_{P_\mathbf{q}}$. We similarly build a term feature vector using the global profile:

$$\mathbf{f_{q*}} = \{t_i | (t_i \in \mathcal{L}(P_{\mathbf{q}*}) \wedge i \leq l) \vee (t_i \in \mathcal{L}(D_m) \wedge i \leq u)\}, \tag{13}$$

where $P_{\mathbf{q}*}$ is the global profile and D_m equals the first m documents returned by $\mathcal{R}_D(\mathcal{C}, \mathbf{q}_{P_{\mathbf{q}*}})$.

We define a function which builds term frequency vectors as:

$$\psi(\mathbf{f_q}, \mathbf{d}_i) = [f_1, \ldots, f_{l+u}]^\top,$$

where the term frequencies $\{f_j | j \leq l + r\}$ are for document $\mathbf{d}_i$ and the terms in $\mathbf{f_q}$. As the profile and returned documents change per iteration, so do the term feature vectors. We define our feature extractor as:

$$\mathbf{\Phi}\left(\left[\{\mathbf{q}, P_\mathbf{q}, \langle s_1, \ldots, s_n \rangle\}\right] \right) = \left[\psi(\mathbf{f_q}, \mathbf{d}_1), \ldots, \psi(\mathbf{f_q}, \mathbf{d}_m)\right]. \tag{14}$$

We also test a hybrid method, which adds the retrieval scores from the current run as an additional feature:

$$\mathbf{\Phi}\left(\left[\{\mathbf{q}, P_{\mathbf{q},}, \langle s_1, \ldots, s_n \rangle\}\right] \right) = \begin{bmatrix} \psi(\mathbf{f_q}, \mathbf{d}_1), \ldots, \psi(\mathbf{f_q}, \mathbf{d}_m) \\ s_1, \ldots, s_m \end{bmatrix}. \tag{15}$$

As an example, suppose there are three documents, i.e. $n = 3$, whose retrieval scores are $\langle 1, 2, 3 \rangle$, the query profile is $P_\mathbf{q} = \{\mathbf{d}\}$, and the term based features for the query are $\mathbf{f_q} = \{t_1, t_2\}$. Furthermore, suppose that for the first document the frequency of t_1 is 1 and that of t_2 is 2, i.e. $\psi(\mathbf{f_q}, \mathbf{d}_1) = [1, 2]^\top$, and that for the second document $\psi(\mathbf{f_q}, \mathbf{d}_2) = [1, 1]^\top$. Assume that we are reordering the top two documents, i.e. $m = 2$.

Our input to the non-hybrid term frequency feature extractor $\mathbf{\Phi}$ will be:

$$\left[\{\mathbf{q}, P_\mathbf{q}, \langle 1, 2, 3 \rangle\}\right],$$

and the output features will be:

$$\begin{bmatrix} 1, 1 \\ 2, 1 \end{bmatrix}.$$

The features are the term frequencies of t_1 and t_2 for each document. The features are $[1, 2]^\top$ for the first document and $[1, 1]^\top$ for the second document. For the hybrid term frequency feature extractor the output features will be:

$$\begin{bmatrix} 1, 1 \\ 2, 1 \\ 1, 2 \end{bmatrix}.$$

Now the features are the term frequencies of t_1 and t_2 for each document as well as the retrieval scores. The features are $[1, 2, 1]^\top$ for the first document and $[1, 1, 2]^\top$ for the second document.

4 Experiments

Our experiments are designed to critically evaluate our research questions using a dataset and user interaction simulation that approximates real world search. We first describe the data, then reiterate our research questions and introduce the experiments we perform to answer them. We evaluate our research questions in the framework of the TREC Legal Track. We calculate trec-eval[15] scores, as well as TREC Legal specific scores,[16] to answer our research questions.

4.1 Data

We use the TREC 2010 Legal Track dataset, which contains:

- a corpus $\mathcal{C}$ with documents $\mathbf{d} \in \mathcal{C}$,
- a set of queries $\mathbf{q} \in \mathcal{Q}$,
- a *seed* set of judged documents for each query, $\{\langle \mathbf{d}, j \rangle | \mathbf{d} \in D_{\mathbf{q}}\}$.

In each pair $\langle \mathbf{d}, j \rangle$ the judgment, denoted by $j \in \{1, -1, 0\}$, indicates that document $\mathbf{d}$ is respectively relevant, non-relevant, or other (judged but an ambiguous judgement). A small number of seed documents have multiple judgements. Previous TREC Legal participants have handled this by considering these documents as unjudged, taking the first judgement as correct, or taking the last as correct. Participants reported that there were no significant effects on retrieval scores [89], therefore, in the case of multiple judgements, we take the last judgment as correct.

The TREC Legal Track addresses search in the context of e-Discovery, a legal procedure in which participants review a document collection to determine which documents are *responsive* to a *request for production*. Relating this to standard information retrieval terminology, *responsive* is analogous to *relevant* and a *request for production* is a *query* (although some call this a *renderer* [49]) that expresses an information need.[17] Throughout we will use *query* in place of *request for production*. As an example of our data set, query number 202 is as follows:

> `All documents or communications that describe discuss refer to report on or relate to the Companys engagement in transactions that the Company characterized as compliant with FAS 140 or its predecessor FAS 125,`

[15] `http://trec.nist.gov/trec_eval/`

[16] `http://durum0.uwaterloo.ca/trec/legal10-results/`

[17] In some cases a request for production can be significantly more complex than a standard query. It may include restrictions to a specific time frame or demarcate *privileged* documents whose distribution should be limited, e.g. because they include discussions between the client and their attorney. The data in the TREC 2010 Legal Track encodes requests for production as paragraphs of text, which look like exceptionally descriptive versions of queries normally encountered in ad-hoc search.

where we have removed all non-alphanumeric characters.

When we apply a frequency based language model to the relevant documents for query 202 the top 5 terms, $t_1 \circ \cdots \circ t_5 | t_i \in \mathcal{L}(P_{\mathbf{q}})$, are:

`Raptor Sara Corp Mary Shackleton`,

we defer discussing issues of term appropriateness to Section 6. To build an expanded query we consider the original query and the feedback terms as two separate queries, and incorporate them into a structured query in Indri syntax, as described in Section 3.2.

Corpus The TREC Legal 2009, 2010, and 2011 corpora are based on the Electronic Discovery Reference Model (EDRM) Enron Dataset v2.[18] All documents that will be evaluated are in the form of email messages or email attachments. To avoid problems with attachments' highly varying length and content format, we ignore attachments which are in the seed set. We preprocess each email message by removing a uniform footer and all non-alphanumeric characters. We additionally convert all text to lowercase and remove any stop-words in a slightly expanded version of the stop-word list from NLTK,[19] a widely used natural language processing toolkit. Given the preprocessed data, we apply the Krovetz stemmer and then build a search index.

Selecting judged documents For each query we have a set of seed documents that were judged relevant or non-relevant. These judgements are unevenly distributed across queries with, on average, more negatively judged documents, as is typical in information retrieval [100]. Figure 5 plots the distribution of judged documents per query. The minimum number of positively judged documents is 19 for query 206, and the maximum is 1006 for query 202. The minimum number of negatively judged documents is 336 for query 206, and the maximum is 1506 for query 205.

In addition to the judged seed documents we also have QRELS judgements, which were used to evaluate results submitted to the TREC 2010 Legal Track. To run experiments that are comparable to the TREC 2010 Legal submissions we must ignore the QRELS, but using them in other experiments allows us to test our method with a larger judged document set and compare the informativeness of arbitrary judged documents to documents in the seed set.

4.2 Evaluation

We define an evaluation function:

$$\mathcal{E}(D, QRELS) = score \in [0, 1], \tag{16}$$

[18] `http://www.edrm.net/resources/data-sets/edrm-enron-email-data-set-v2`
[19] `http://www.nltk.org/`

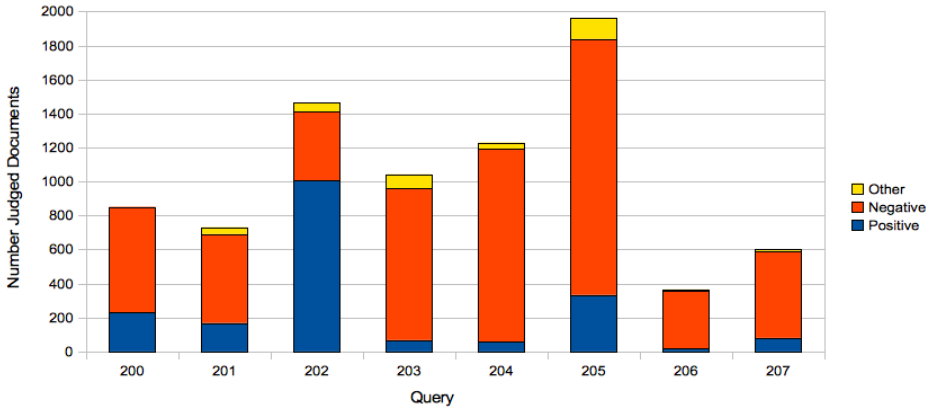


Fig. 5. Distribution of judged documents per query. Those marked "Positive" are relevant documents, "Negative" are non-relevant documents, and "Other" are judged documents with an unknown judgement.

which returns the score of the ordering D, based on the judgements in $QRELS$, as a rational number between 0 and 1.[20] For our evaluation we use standard retrieval metrics as well as metrics specific to TREC Legal. The standard metrics of mean average precision (MAP) and normalized discounted cumulative gain (NDCG) show fine grained differences in retrieval performance [11], and are therefore better suited to evaluating our research questions. We additionally evaluate with area under the curve (AUC), which is one of the two metrics used in TREC Legal. When using AUC we can compare our results to those published in the proceedings of the TREC 2010 Legal Track.

4.3 Experimental setup

To evaluate our research questions we incrementally build judged document profiles by considering documents judged with respect to only a single query and those judged for multiple queries. We use the query-specific and global profiles to perform query expansion, which provides baseline retrieval scores. We then use learning to rank to reorder. By comparing scores before and after reordering when using different profiles, feature spaces, and sampling methods, we can evaluate all of our research questions.

Because our learning to rank method cannot compute a valid distance measure until we have at least one positive and negative instance (noted by Xu and Akella [106] for a similar task), at each iteration we expand the query profile by adding one relevant and one non-relevant document. Therefore, at iteration t there will be $2t$ judged documents, with t relevant and t non-relevant documents. We build a set of positive expansion terms by taking the top 5 terms

[20] When using area under the curve (AUC) the range is 0 to 100, i.e. $\mathcal{E}(D, QRELS) = score \in [0, 100]$.

according to frequency (after stop-word removal) from the relevant documents. In preliminary tests, retrieving documents using queries expanded with 5 terms outperformed retrieval using the original queries, and was not significantly different than retrieval scores when expanding queries with 10, 15, or 20 terms.

We run our experiments for each query $\mathbf{q}$ and also calculate the average retrieval score over all queries. Below we pair each research question with the experiment we perform to answer it.

Research Question 1 *Does global feedback from queries related to the user's general information need improve a search system's ability to re-rank for new queries?*

Experiment 1 *Calculate retrieval scores when applying learning to rank using a preference pair set built from a query-specific profile and a global profile.*

We will compare the query-specific profile and global profile when using the different feature sets and sampling methods. Given a query profile $P_{\mathbf{q}}$, specific to query $\mathbf{q}$, and a global profile $P_{\mathbf{q}*}$, also for query $\mathbf{q}$, we evaluate:

$$\mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}}}) \stackrel{?}{>} \mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}*}}),^{21}$$

to compare the effectiveness of reordering with these two types of profiles.

The query-specific profile only includes judged documents from the current query, while the global profile additionally includes one judged document from all other queries. We add only one judged document from other queries so that the amount of data for the current query is at least as much as that for any other query. On the first iteration the current query and all other queries will each have one judged document. On all subsequent iterations the current query will have more judged documents than any other query. In preliminary experiments we evaluated adding a larger number of the judged documents for other queries, including all of the judged documents for other queries. These larger global profiles significantly underperform a global profile that adds only a single document.

Research Question 2 *How is a search system's ability to learn to rank results affected by the number of judged documents?*

Experiment 2 *Calculate the change in retrieval scores as the size of our document profile increases.*

Where $\mathcal{S}_{\mathcal{P}_{\mathbf{q}}^{(i)}}$ is the corpus when reordering with preference pairs $\mathcal{P}_{\mathbf{q}}^{(i)}$, for query $\mathbf{q}$ after i iterations, we plot:

$$\{\mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}}^{(1)}}), \mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}}^{(2)}}), \dots\},$$

[21] For the sake of clarity here, and throughout this section, we drop the $QRELS$ argument in $\mathcal{E}$ and drop the function arguments for $\mathcal{S}_{\mathcal{P}}(D, \mathbf{q})$. The context makes clear that, in $\mathcal{S}_{\mathcal{P}}$, D is as created in Algorithm 2 and $\mathbf{q}$ is irrelevant or the subscript of $\mathcal{P}$.

and evaluate:

$$\mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}}^{(i)}}) \stackrel{?}{<} \mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}}^{(i+1)}}),$$

where each iteration $j > i$ has more judged documents. If the answer to research question 2 is affirmative we expect that, on average, iteration j will have higher retrieval scores than iteration i.

Research Question 3 *Which sampling method chooses documents that are most beneficial to retrieval scores when using learning to rank?*

Experiment 3 *Calculate retrieval scores when using different sampling methods.*

We will compare retrieval scores for our sampling methods when using different profiles and feature spaces.

Research Question 4 *Which feature space most improves retrieval scores when applying learning to rank?*

Experiment 4 *Calculate retrieval scores when using cumulative, constant, and term based feature spaces.*

Before the first iteration we do not have any judged documents and will not reorder. After the first iteration, and for future iterations, we have a judged documents set with which to expand queries, build features, and learn to rank.

Parameter settings In all per-iteration experiments we will add one judged relevant and one judged non-relevant document per iteration. We run each experiment for 20 iterations, i.e. from a total of 0 to 20 each of added relevant and non-relevant documents. We chose a maximum of 20 iterations because the smallest number of judged seed documents for a query is 19, for query 206's relevant documents, and therefore after 19 iterations query 206 cannot sample any more relevant seed documents.

In query expansion we set $\alpha = 0.8$, placing greater weight on the original query. We chose this value because in preliminary experiments it produced results that, on average, outperformed the baseline without expansion. Optimizing this value is not important to our research questions because it remains constant across our different experiments; it is the comparative retrieval scores between variations of our method that we are most interested in.

When using cumulative features we take a maximum of 10 features for the exploitative and random seed sampling methods, and 15 features when sampling from all documents. During our retrieval runs not all documents are returned, and documents which are not returned for a particular iteration have neither a retrieval score nor a feature for that iteration. The maximum feature values of 10 and 15 are the minimum number of features that ensure each document which we reorder has at least one feature, and that we can therefore reorder this document in learning to rank.

We build constant features by using the retrieval scores of each document when expanding the base query with 5, 10, 15, and 20 top terms from the set of positive terms. This provides us with four features, one from the retrieval scores in each expansion run. For both cumulative and constant features we reorder the top 10,000 documents. This was decided based on computational constraints and because preliminary experiments showed that reordering fewer documents did not perform significantly better.

When using term-frequency features we take the top 2000 terms from judged documents, $l = 2000$, and the top 500 terms from the documents we are reordering, $r = 500$. We experimented with a smaller number of terms and chose these values because they are large enough to effectively represent term feature-space and small enough that they do not overwhelm our computing resources. Other than increasing resource usage, there do not appear to be any negative effects of increasing the number of term frequency features, which is what we would expect if our learning to rank algorithm is working properly. For term-frequency features we reorder the top 1000 documents. This was again decide because of computational constraints and because preliminary experiments showed that reordering fewer documents did not perform significantly better.

4.4 Summary

Table 1 summarizes the experiments we perform and the comparisons we make. For each combination of feature space and sampling strategy we compare the

Table 1. Summary of experiments. Sampling method is shown by row and feature space by column.

Sampling \ Features	Cumulative	Constant	Term frequency
Exploitative	Query-specific Global	Query-specific Global	Query-specific Global
Random seeds	Query-specific Global	Query-specific Global	Query-specific Global
Random all	Query-specific Global	Query-specific Global	Query-specific Global

global and query-specific profiles as we add judged documents to these profiles. We also compare cumulative, constant, and term frequency features. Finally, we compare exploitative sampling, random sampling, and sampling from all documents.

5 Results

We present our results so as to address each research question in turn. We first compare results when using the baseline with no expansion, query expansion, and learning to rank using the query-specific and the global profile. We then present per iteration results for one of the best performing method variations: constant features, the query-specific profile, and randomly sampling from the seed documents.

We next compare the different sampling methods, and then the different feature spaces. Lastly, we compare our results to those in the TREC 2010 Legal Track. All statistical significance tests are done using a paired student's t-test and evaluate a single hypothesis.

5.1 Learning with query-specific and global profiles

Table 2 presents MAP and NDCG scores for the last of 20 iterations and all method combinations as depicted in Table 1. We compare all columns for each row to evaluate the profiles in relation each other, the baseline, and query expansion. Scores are averaged over all queries.

Table 2. Scores for the last of 20 iterations averaged over all queries. The top section shows scores when using cumulative features, the middle section when using constant features, and the bottom section when using term frequency features. Scores marked $^{\triangle}$ are statistically significant at the 0.01 level when compared to the baseline scores. Scores marked $^{\blacktriangle}$ are statistically significant at the 0.001 level when compared to the baseline scores.

Features	Sampling	Metric	Baseline	Expansion	Query	Global
Cumulative	Exploitative	MAP	0.0674	**0.108**	0.0852	0.0607
		NDCG	0.565	$^{\triangle}$**0.630**	0.598	0.550
	Random seeds	MAP	0.0674	$^{\triangle}$**0.130**	$^{\triangle}$0.121	0.0637
		NDCG	0.565	$^{\triangle}$**0.646**	$^{\triangle}$0.638	0.561
	Random complete	MAP	0.0674	**0.0960**	0.0873	0.0512
		NDCG	0.565	$^{\triangle}$**0.625**	0.609	0.544
Constant	Exploitative	MAP	0.0674	**0.108**	0.0975	0.0896
		NDCG	0.565	$^{\triangle}$**0.630**	0.613	0.600
	Random seeds	MAP	0.0674	$^{\triangle}$0.130	$^{\triangle}$**0.137**	0.124
		NDCG	0.565	$^{\triangle}$0.646	$^{\blacktriangle}$**0.652**	0.641
	Random complete	MAP	0.0674	0.0960	**0.109**	0.0896
		NDCG	0.565	$^{\triangle}$0.625	$^{\triangle}$**0.637**	$^{\triangle}$0.626
Term frequency	Exploitative	MAP	0.0674	0.108	0.100	**0.114**
		NDCG	0.565	**0.630**	0.612	0.626
	Random seeds	MAP	0.0674	0.0886	0.0942	0.0856
		NDCG	0.565	**0.609**	0.607	0.596
	Random complete	MAP	0.0674	0.134	**0.137**	0.0969
		NDCG	0.565	$^{\blacktriangle}$**0.648**	$^{\triangle}$0.646	0.598

In the highlighted row in Table 2 we see that our method significantly outperforms the baseline and outperforms query expansion for both MAP and NDCG score when using constant features and randomly sampling seed documents. This shows that, even with a small number of judged documents, re-ranking using our method can improve scores beyond those achievable when using only query expansion. This answers our overarching question concerning how to use the interactive nature of search:

> *By combining traditional query expansion with learning to rank, a search system can use the interactive nature of search to better serve the user's needs.*

In the following we present detailed results for all combinations of feature spaces and sampling strategies when learning to rank from judged documents in the query-specific profile and the global profile.

Cumulative retrieval score features Using cumulative retrieval score features the global profile performs very poorly, its maximum scores are when sampling randomly from the seed documents. For this sampling strategy, the global profile's MAP score of 0.0637 and NDCG score of 0.561 underperform all other methods, including the baseline; which has a MAP score of 0.0674 and an NDCG score of 0.565. Like the global profile, both the query-specific profile and query expansion perform best when randomly sampling seed documents. For this sampling strategy, the query-specific profile has a MAP score of 0.121 and an NDCG score of 0.638. Query expansion outperforms the query-specific profile, which has a MAP score of 0.130 and an NDCG score of 0.646. Across sampling strategies the query-specific profile consistently outperforms the baseline, but also consistently underperforms query expansion.

Constant retrieval score features When using constant retrieval score features the baseline underperforms all other methods across all sampling strategies. When randomly sampling seed documents the query-specific profile has a MAP score of 0.137 and an NDCG score of 0.652, outperforming all other methods including query expansion; which has a MAP score of 0.130 and an NDCG score of 0.646. Reordering with the global profile has a MAP score of 0.124 and an NDCG score of 0.641, when randomly sampling seed documents. The global profile outperforms the baseline but underperforms query expansion and the query-specific profile.

Term frequency features When using term frequency features the comparative performance of learning to rank and query expansion varies more per sampling strategy and retrieval metric than when using the other feature spaces. When sampling using exploitative sampling the global profile has a MAP score of 0.114 and an NDCG score of 0.626; this outperforms the query expansion

MAP score of 0.108, but underperforms its NDCG score of 0.630. On both metrics the global profile outperforms the query-specific profile, which has a MAP score of 0.100 and an NDCG score of 0.612.

When sampling from random seed documents, the query-specific profile has a MAP score of 0.0942, outperforming the query expansion MAP score of 0.0856. However, it has a NDCG of 0.607; underperforming the query expansion NDCG score of 0.609. Similarly, when sampling from all judged documents, the query-specific profile has a MAP score of 0.137, outperforming the query expansion score of 0.134; but it has an NDCG score of 0.646, slightly underperforming the query expansion score of 0.648.

5.2 Performance change over iterations

Table 3 shows the per iteration change in MAP scores for iterations 2 and 18-20, when using constant features; learning to rank with the query-specific profile; and randomly sampling from the seed documents. The left hand side shows the change from iteration to iteration and the right hand side shows the change from the first iteration to the current iteration, both also show these changes as a percentage. The final row shows the sum per iteration of all changes from iteration to iteration, which is a 26% increase in MAP score. When we look at the second to last row, which shows the final iteration, we see that the change from the first iteration is a substantial increase of 23%.

Table 3. Constant features, query-specific profile, and random seed sampling. Change in MAP scores per iteration from the last iteration and from the first iteration, as well as the percent change for both. The final row shows the sum of the actual and percent changes iteration to iteration.

Iteration	Change from last iteration	Percent	Change from iteration 1	Percent
2	-0.0054	-4.8	-0.0054	-4.8
⋮	⋮	⋮	⋮	⋮
18	-0.00011	-0.085	0.022	20
19	0.0020	1.5	0.024	22
20	0.0015	1.1	0.026	23
sum	0.026	26	-	-

5.3 Comparison across sampling strategies

Table 4 compares MAP scores across the different sampling strategies when using query expansion and reordering with the query-specific and global profile. When using either cumulative or constant features, and for both document profiles, randomly sampling from the seed documents outperforms exploitative and randomly sampling from all documents. When using term frequency features the

Table 4. Cumulative features (top), constant features (middle), and term frequency features (bottom) across sampling strategies. We average MAP scores for the last iteration over all queries.

Features	Method	Exploitative	Random seeds	Random all
Cumulative	Query-specific	0.0852	**0.121**	0.0873
	Global	0.0607	**0.0637**	0.0512
Constant	Query-specific	0.0975	**0.137**	0.109
	Global	0.0896	**0.124**	0.0896
Term	Query-specific	0.100	0.0942	**0.137**
frequency	Global	**0.114**	0.0856	0.0969

results are less consistent. As compared to other sampling strategies, randomly sampling from all documents outperforms when learning to rank with the query-specific profile, while exploitative sampling outperforms when learning to rank with the global profile.

5.4 Comparison across feature spaces

Table 5 compares the MAP scores across feature spaces when learning to rank with the query-specific and global profile. The top portion of Table 5 presents

Table 5. Random sampling seeds (top) and exploitative sampling (bottom) across all feature extraction methods. We average MAP scores for the last iteration over all queries.

Sampling	Method	Constant	Cumulative	Terms
Random seeds	Query-specific	**0.137**	0.121	0.0942
	Global	**0.124**	0.0637	0.0856
Exploitative	Query-specific	0.0975	0.0852	**0.100**
	Global	0.0896	0.0607	**0.114**

MAP scores when randomly sampling from the seed documents and the bottom portion when using exploitative sampling. We see that for randomly sampling and both document profiles, constant features outperform the other features spaces. While for exploitative sampling and both document profiles, term frequency features outperform.

5.5 Comparison to TREC 2010 Legal

Table 6 presents the maximum, average, and minimum AUC scores from participants in the TREC 2010 Legal Track. The far right of Table 6 presents scores for the best performing of comparable experiments in this work (ones which ignore the QRELS file). These scores are when using constant features, the query-specific profile, and exploitative sampling. Our method has an average AUC

Table 6. Comparison of area under the curve (AUC) scores for the TREC 2010 Legal Track maximum, average, and minimum scores with our method when using constant features, the query-specific profile, and exploitative sampling. TREC Max shows scores for the run with the maximum average, similarly TREC Min shows scores for the run with the minimum average.

Query	TREC Max	TREC Avg	TREC Min	This Research
200	74.6	72.0	63.5	**84.1**
201	**94.7**	74.9	46.2	87.3
202	**99.4**	84.1	44.5	62.0
203	**95.1**	78.5	49.5	74.8
204	76.2	71.9	58.0	**85.9**
205	**75.3**	71.1	52.4	74.8
206	83.1	73.3	**87.3**	82.2
207	**95.3**	61.4	44.6	94.0
avg	**86.7**	71.6	55.8	80.6

score of 80.6, which is near the mode score and would have been ranked 10th best, with 9 systems above and 11 below. Although our score is less than the maximum of 86.7, it outperforms the average of 71.6 by 13% and the minimum score of 55.8 by 44%.

When we examine results query by query, we see that our method scores 84.1 and 85.9 on queries 200 and 204 respectively, outperforming the respective maximum scores of 74.6 and 76.2. On queries 205, 206, and 207 our method is also competitive with the maximum scores, underperforming by only 0.66%, 1.1%, and 1.4% respectively. Overall, our results are clearly competitive with those in TREC 2010 Legal.

The performance of our method, when compared to the TREC 2010 Legal participants, varies between queries. We hypothesize that one reason for this may be that the judged documents for some queries are more amenable to learning to rank and query expansion than the judged documents for other queries. This would be true if the most frequent terms in one query's set of judged documents are better at distinguishing between relevant and non-relevant documents than those in another query's. The terms would therefore be more useful in query expansion and in creating features for use in learning to rank.

For example, the query expansion terms from relevant documents for query 206, for which our system outperforms the maximum TREC 2010 Legal AUC score, contain the distinctive proper noun `Dynegy`. However the query expansion terms from relevant documents for query 202, for which our system underperforms the maximum and average TREC 2010 Legal AUC scores, do not contain similarly distinctive terms. We expect that, because the high frequency terms are more distinctive, our system is able to build a more representative feature space for query 206 than for query 202.

6 Discussion

We comment in turn on how our results pertain to each of our research questions and draw conclusions from our data to answer each question. We also discuss interesting nuances or exceptions to the general answers to our research questions and present other conclusions based on our results.

6.1 Research Question 1: The query-specific profile outperforms the global profile

Our first research question asks whether learning to rank with a query-specific profile of judged documents will perform better than learning to rank with a global profile of judged documents. When using either cumulative or constant features, reordering with the query-specific profile outperforms the global profile. However, the global profile outperforms the query-specific profile when using term frequency features and exploitative sampling.

To capture broader user interests and find terms that are relevant for the general context of the user's queries, we build a global profile that includes documents relevant to other queries in addition to those relevant to the current query. If the user's information needs are similar enough, the global profile may be able to discover patterns that are useful across all queries. Our results show that when using term frequency features and exploitative sampling the global profile performs better than the query-specific profile.

When examining the term frequency features and exploitative sampling results per query, we see that the global profile outperforms the query-specific profile by an especially large margin on query 200. Query 200 contains judged documents selected in a different—and pontentially poorer with regards to learing—manner than queries 201-207 (we discuss this in more detail below in Section 6.4). We expect that by including highly ranked relevant documents that were selected in a different manner, i.e. the relevant documents for queries 201-207, the global profile builds a more representative set of terms, which are more useful as features in learning to rank than documents for only query 200.

In contrast, for the cumulative and constant feature spaces, the query-specific profile outperforms the global profile. We expect that this is because the relevant documents for queries other than the current query are not necessarily relevant to the current query, and similarly for non-relevant documents. It is even possible that these relevant global profile documents would be judged non-relevant for the current query had the user been given the opportunity.

The cumulative and constant feature spaces use a small number of features when compared to the term frequency feature space. To demonstrate this, we consider a document that is not relevant for the current query but appears in the global profile. Due to the large number of shared document terms and the uniqueness of retrieval scores, it is probable that there is greater discord between the retrieval scores of this document and the retrieval scores of query-specific relevant documents, than there is between this document's term distribution and the term distributions of query-specific relevant documents. Therefore it is

reasonable to expect that learning from the retrieval scores of a global profile document that is not relevant for the current query, can harm performance substantially more than learning from the contents of that document, e.g. when using term frequency features. For the cumulative and constant feature spaces it appears that a broader document profile has a more difficult time capturing characteristics common to all the user's queries.

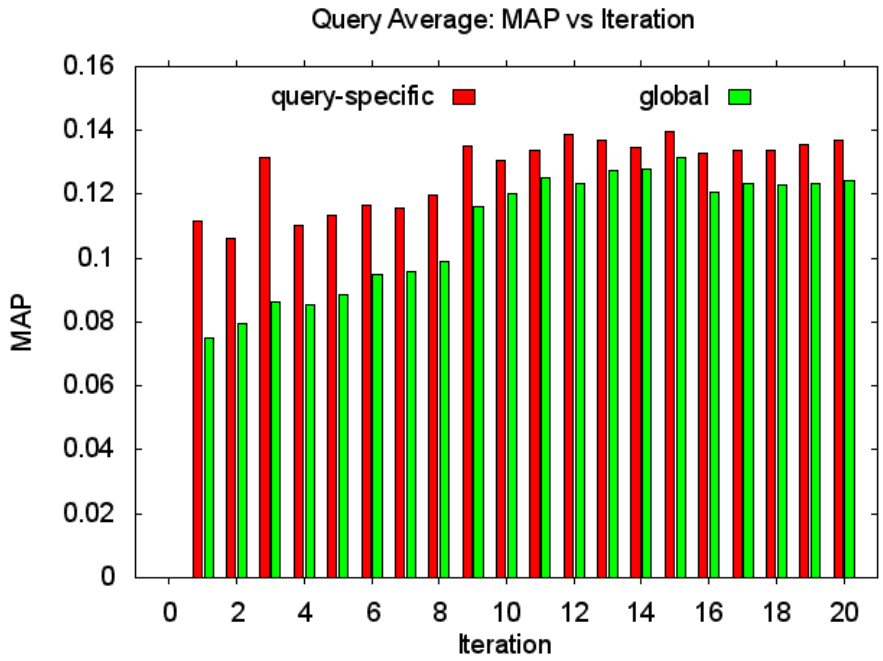
Figure 6(a) shows the average score for each of 20 iterations after reordering with the query-specific and global profile when using the constant method and randomly sampling from seed documents. Although the difference in performance is not significant (expected given the small number of queries), the 10% improvement in MAP score suggests that with more data it may be. Furthermore, reordering with the query-specific profile consistently performs better than reordering with the global profile. Compared to the MAP scores of the global profile, the query-specific profile provides an 8% improvement for exploitative sampling and constant features, a 40% improvement for exploitative sampling and cumulative features, and a 90% improvement for random seed sampling and cumulative features. We therefore conclude that, in the majority of cases, learning to rank with the query-specific profile outperforms the global profile:

$$\mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}}}) > \mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}*}}).$$

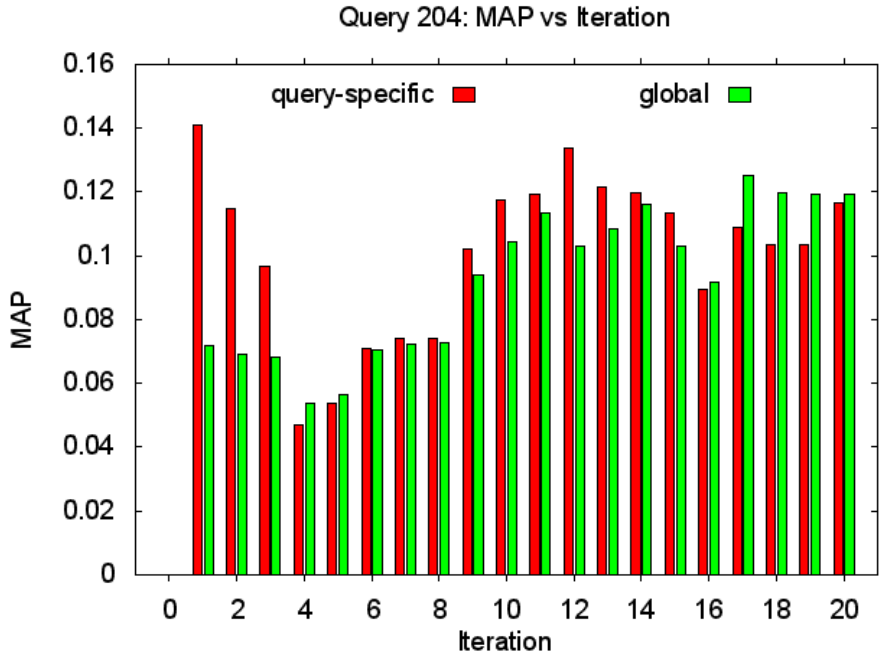
In Figure 6(a) we can also see that retrieval scores increase for both profiles after more iterations, but more so for the global profile than the query-specific profile. As more iterations pass the proportion of documents in the global profile that are for other queries decreases relative to those for the current query. We expect that this is a factor in increasing global profile scores for later iterations, and explains why per iteration scores increase more for the global profile than for the query-specific profile.

The global profile outperforms the query-specific profile for query 204 When we look at the scores per query for random seed sampling and constant features, we see that the global profile underperforms for all queries except query 204, and then only in later iterations. Figure 6(b) shows the MAP scores per iteration for query 204. On iteration 16 the global profile begins to outperform the query-specific profile. On this iteration the MAP score for the query-specific profile precipitously decreases to 0.0895, from 0.113 on the previous iteration. Similarly, the query expansion scores (not plotted) decrease steeply to 0.0499 on iteration 16, from 0.0919 on the previous iteration. We hypothesize that an added document judgement has negatively affected the expansion terms and led to these sharp decreases.

The relevant document sampled on iteration 16 was an email concerning the merger between Calpine Energy Services and Calpine Power. Query 204 requests documents referring to the alteration or destruction of data, and this email is relevant because it mentions that the name of all Calpine Power records will be changed to "Calpine Energy," due to the merger. Because of forwarding and replying the new email contains four quoted emails.



(a) Random seeds, constant features; comparing the query-specific and global profiles. Average MAP score over all queries, scores are per iteration.



(b) Random seeds, constant features; comparing the query-specific and global profiles. MAP scores for query 204 per iteration.

Fig. 6. Random seeds, constant features; comparing the query-specific and global profiles.

The term `Corp` increases in frequency because it is included in the recipient line of these emails, as: `Adnan Patel/Corp/Enron@ENRON`. This is reflected in the expansion terms taken from relevant documents, which on iteration 15 are:

`2000 Outlook ENRON_DEVELOPMENT Communications email`.

After sampling the new document on iteration 16, the expansion terms become:

`2000 Outlook ENRON_DEVELOPMENT Communications Corp`,

where the term `email` has been replaced by the term `Corp`. Apparently, this causes query expansion to retrieve a large number of non-relevant documents, and thereby reduces the MAP score. We could eliminate the term `Corp` by using an alternative preprocessing algorithm, but this reduction in information would cause other problems.[22]

By using the global profile we reduce the negative effects of this new judged document without requiring any extra preprocessing. When using the global profile, the new judged document on iteration 16 decreases the MAP score by only 11%, to 0.0918 from 0.103 on iteration 15. This decrease in score is much less than the 21% decrease when reordering with the query-specific profile, or the 46% decrease when using query expansion.

To further examine the effect that different profiles have on reordering, Table 7 compares the Kendall's τ rank correlation coefficient of the query-specific and global profiles on iterations 15 and later. Kendall's τ is in the range $[-1, 1]$

Table 7. Kendall's τ rank correlation coefficient for query 204 on iteration 15 through 20, as well as the numeric and percentage difference in correlation: $\tau_{query\text{-}specific} - \tau_{global}$.

Iteration	Query-specific	Global	Difference
15	0.648	0.643	+0.005
16	0.642	0.644	−0.002
17	0.620	0.641	−0.021
18	0.629	0.639	−0.010
19	0.628	0.636	−0.008
20	0.650	0.636	+0.140

[22] For example, we could ignore the recipient line or not split terms on the "/" character, but then we would lose (or obfuscate) information about entities, which is often very informative. As an example of the recipient line's importance, consider that the expansion terms on iteration 20 for query 201 include `Shackleton`, an employee last name appearing in many email recipient lines. (Entity names also occur in salutations, introductions, and other locations in emails. To fully quantify the importance of the recipient line we must run additional tests varying their exclusion, which is beyond the scope of this research.) We could compensate for this loss of information by separately modeling the graph of email communications, however this solution would only apply to emails or other documents accompanied by a graph.

with 1 indicating perfect agreement in order, −1 perfect inversion, and 0 no association. Between iterations 15 and 16, Kendall's τ decreases by 0.9% for the query-specific profile, showing that the order agrees less after we sample the new judged document. On the other hand, when using the global profile, Kendall's τ increases slightly, by 0.1%, showing that the orders agree more. This suggests that the effect on reordering of adding this new document to the query-specific profile is greater than—and contrary to—that of adding it to the global profile. This is what we expect because the new document is a greater proportion of the total number of documents in the query-specific profile than it is in the global profile. The change in rank correlation is likely the cause for the decrease in scores when using the query-specific profile.

Returning to Figure 6(b), we see that for query 204 after iteration 16, the query-specific profile continues to underperform the global profile, especially from iteration 17 through 19, and only slightly for iteration 20. The right hand column in Table 7 shows the difference in Kendall's τ between the two profiles: τ for the query-specific profile minus τ for the global profile. We see that for iteration 17 through 19, when the query-specific profile underperforms, Kendall's τ of the global profile is greater than that of the query-specific profile, and vice-versa for iteration 20.

Figure 7 compares the percent change in Kendall's τ to the percent change in MAP score between the query-specific and global profiles from iteration 15 through 20. When both bars are negative it shows that both Kendall's τ and the

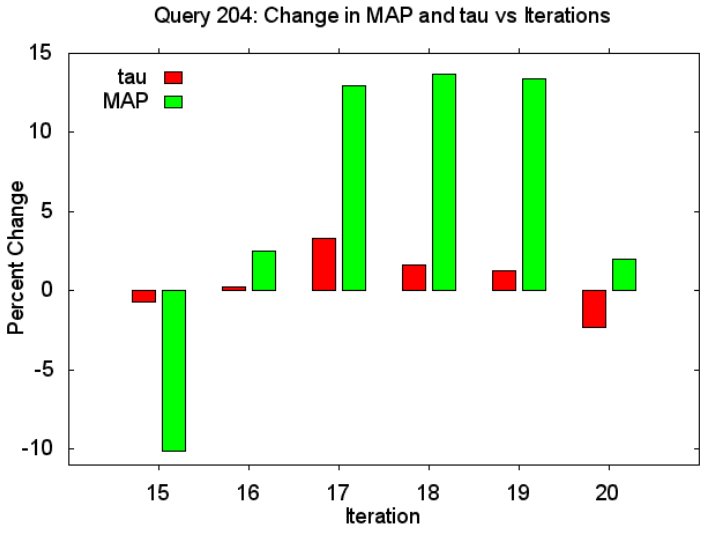


Fig. 7. Percent change in Kendall's τ and MAP score between the query-specific and global profile from iteration 15 through 20.

MAP score of the global profile are lower than that of the query-specific profile, and vice-versa when both bars are positive. Excluding iteration 20, we see that the change in Kendall's τ correlates with the change in MAP score: if Kendall's τ is higher for the global profile than for the query-specific profile, then so is MAP score. We conclude that, for query 204, by learning to rank from more diverse documents the global profile is changing the order less—it has a higher Kendall's τ—and is therefore affected less by a judged document which hurts performance—it has a higher MAP score.

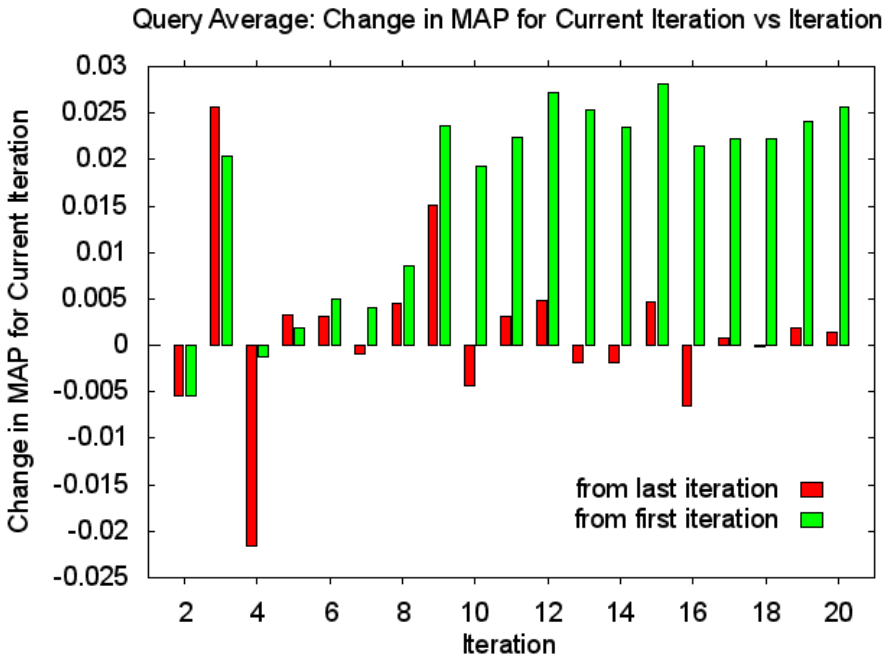
6.2 Research Question 2: More judged documents help learning

Across all methods, we see that as more judged documents are used for learning to rank the ability to improve scores increases. Figure 8(a) shows the change in MAP score from iteration-to-iteration and from the first iteration to the current iteration; when using constant features, the query-specific profile, and random sampling from seed documents. This plot shows that, from iteration-to-iteration, there are a greater number of iterations for which MAP score increases than decreases. The change in performance from iteration-to-iteration is rather unstable. This is expected because each individual document could be an outlier which substantially changes performance. However the sum of the changes in scores is 0.026, showing that—on average—MAP scores increase from iteration-to-iteration.

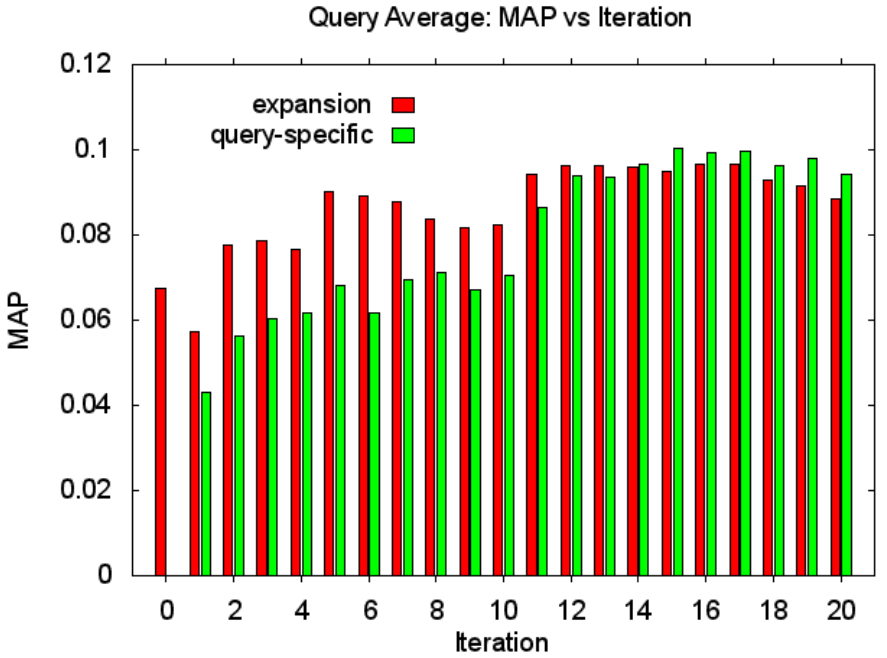
When using learning to rank we expect that by having a more representative feature space later iterations are able to better learn a model for reordering documents and thereby improve scores. How this larger set of documents translates into more representative features is most intuitive in the term frequency feature space, where having more instances implies a larger set of terms, which is more representative of the term-space for our relevant and non-relevant documents. In the constant and cumulative feature spaces we build expanded queries based upon term frequency. Therefore the same reasoning applies: more judged documents lead to a more representative term-space and a better estimate of the most representative terms which are used in the expanded queries.

Figure 8(a) also shows the change in MAP score between the current iteration and the first iteration, which is a cumulative version of the iteration-to-iteration change. We see that there are no large improvements in performance for the first half of the iterations. Then, after around half of the iterations have passed, MAP scores greatly increase, which is also shown by the large iteration-to-iteration increase in MAP score.

When examining the per query MAP scores it appears that the large increase in average MAP score, seen about midway through the iterations, is because each query requires a different number of judged documents in order for learning to rank to be effective. After about half of the total number of iterations have passed there are enough judged documents for learning to rank to be effective for all queries. In the retrieval score based methods (such as the constant method examined here), this improvement relates to the terms in judged documents



(a) Change in average MAP score from the previous (labelled "last") iteration and from the first iteration. MAP scores are for constant features, the query profile, and randomly sampling from the seed documents.



(b) Sampling random seed documents, comparison between query expansion and re-ordering with the query-specific profile using term frequency features. Average MAP score over all queries, scores are per iteration.

Fig. 8.

being representative of the relevant documents (as discussed above) and, because of this, useful in query expansion.

As an example of this, from iteration 8 to 9 there is a significant increase in query 202's score. The query expansion terms change from:

```
Communications 2000 said Agreement Corp
```

on iteration 8 to:

```
Communications 2000 said Agreement Hawaii
```

on iteration 9. Between these iterations the term `Corp` is replaced by the term `Hawaii`. The term `Corp` is likely uninformative, as suggested by our analysis of query 204 in Section 6.1. We expect the new term `Hawaii` to be more representative of relevant documents, more useful in query expansion, and to lead to more useful features in learning to rank on this and subsequent iterations.

After enough judged documents have been collected for each query and their MAP scores have increased beyond some threshold, the average MAP score remains more or less stable. After the jump in MAP score, shown by the change from the first iteration's MAP score at around iteration 9 in Figure 8(a), the MAP score remains about 25% greater than the MAP score in the first iteration. This indicates that there may be a plateau after we've added a modest number of judged documents, although we will have to evaluate our method with additional data to confirm this.

Scores increase even more significantly for term frequency features When compared to constant features, term frequency features increase scores even more significantly as we add judged documents. Figure 8(b) plots the MAP scores per iteration for query expansion when sampling random seed documents, and learning to rank with the query-specific profile using term frequency features. We see that reordering significantly improves MAP scores as we add judged documents, while this improvement is much less significant for query expansion. Although there are again too few queries and iterations to draw any strong conclusions, there appear to be two plateaus in performance improvement. The first plateau is for the first half of the iterations, during which MAP scores average 0.0831, then performance improves and there is another plateau for the second half of the iterations, during which MAP scores average 0.0945, an increase of 14%.

Although we only have data from 8 queries, the difference in MAP score between iteration 2 and 20 is statistically significant at the 0.05 level. Between these iterations the average MAP score increases by 120%, which suggests that with more data the increase in MAP score may be even more significant. Based on the various examples provided above, we conclude that on average:

$$\mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}}^{(i)}}) < \mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}}^{(i+1)}}) \qquad \mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}}^{(1)}}) < \mathcal{E}(\mathcal{S}_{\mathcal{P}_{\mathbf{q}}^{(20)}}),$$

the subsequent iteration, $i + 1$, has a higher retrieval score than the previous iteration, i; and the final iteration, 20, has a higher retrieval score than the first iteration, 1.

6.3 Research Question 3: Randomly sampling from seed documents outperforms other sampling strategies

For cumulative and constant features, randomly sampling from seed documents outperforms both randomly sampling from all documents and exploitative sampling. Figure 9 plots the change in MAP scores, for all feature spaces and profiles, when randomly sampling from seed documents compared to sampling from all documents, and exploitative sampling. When we compare the percent change be-

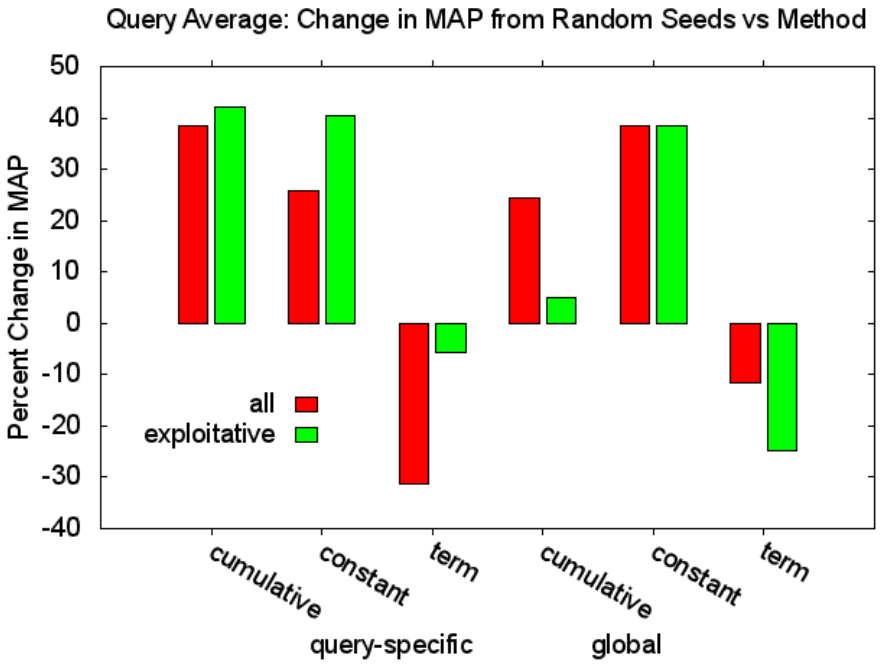


Fig. 9. Comparison of the change in MAP scores from randomly sampling seed documents to sampling from all documents and exploitative sampling. The x-axis labels the feature spaces and the learning to rank profile. For example, the far left hand shows that for the query-specific profile and cumulative futures, sampling random seed documents improves the MAP score by 39% when compared to sampling all documents, and sampling random seed documents improves the MAP score by 42% when compared to exploitative sampling.

tween sampling seed documents and sampling all documents, we see that MAP scores increase by 39% and 26% respectively for cumulative and constant features, when using the query-specific profile; and by 24% and 38% respectively when using the global profile.

The organizers of TREC Legal specifically chose the documents which make up the seed set because they are informative, or because they clearly fall into the

category of relevant or non-relevant—and *not* both categories. When sampling from all documents we may be choosing documents which are (non-)relevant but uninformative and not exemplary of a (non-)relevant document. We would expect that if the documents we are sampling were specifically chosen to inform us about the corpus, they will be more effective—especially when used in combination with each other—than samples that may or may not be informative. Our results show that sampling seed documents outperforms sampling from all documents, confirming this intuition.

When we compare random seed to exploitative sampling, scores increase by 42% for cumulative features and 41% for constant features—when using the query-specific profile. When using the global profile, scores increase by 5% for cumulative features and 38% for constant features. Although the increase in scores is not statistically significant, percentage improvements this large suggest that sampling from random seeds should be the preferred sampling method for constant and cumulative features, and that given a larger set of queries the difference may be statistically significant—excepting the global profile with cumulative features.

Random sampling underperforms other sampling strategies when using term frequency features When using term frequency features there is no sampling method that appears significantly better, but randomly sampling from the seed documents consistently underperforms. When using the query-specific profile, randomly sampling from all documents outperforms other sampling strategies: it improves 49% over random seed document sampling and 46% over exploitative sampling. Again, although none of these changes are statistically significant, the large increase suggests that randomly sampling from all documents is the best strategy when using the query-specific profile.

However, for the global profile, exploitative sampling outperforms other sampling strategies, it improves 5% over randomly sampling from all documents and 17% over randomly sampling from the seed documents. These are relatively small changes and therefore it remains inconclusive which sampling strategy is best for the term frequency feature space. In Section 6.4 we investigate how performance is correlated between feature space and sampling method.

The sampling method used affects performance over time The sampling method that we use can significantly change the effect of adding more judged documents to our profiles. Figure 10 plots the MAP scores per iteration when sampling from the seed documents and from all documents using cumulative features and the query-specific profile. In agreement with the percentage improvement analysis performed above, Figure 10 provides further evidence that sampling from seed documents outperforms sampling from all documents.

Over time, MAP scores improve when sampling from seed documents while they remain more or less constant when sampling from all documents. As described in our previous discussion of the contents of the seed set, this is the expected performance because seed documents are more informative than the

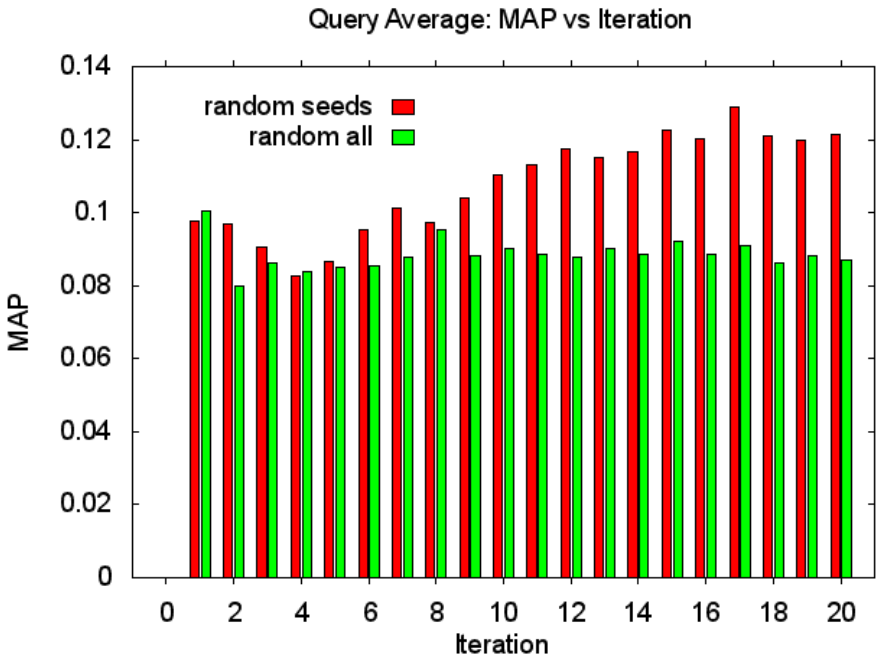


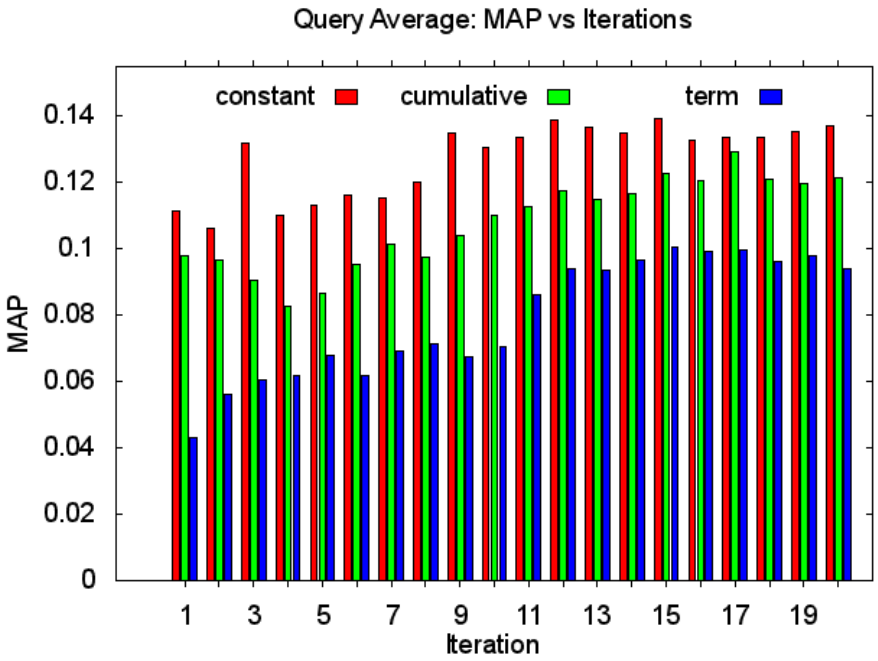
Fig. 10. MAP scores when randomly sampling from seed documents and from all documents using the cumulative feature space and the query-specific profile.

average document. Furthermore, the gains achievable when combining many definitely informative documents are greater than the gains when combining many somewhat informative documents. This shows that when we perform active learning it is important to spend the time needed to determine which documents will be informative if they are judged.

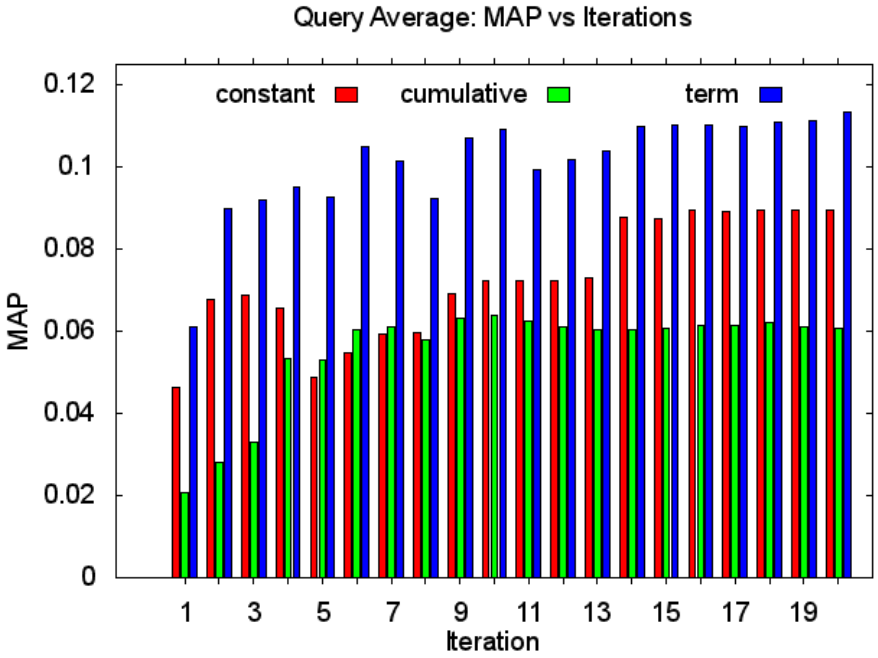
6.4 Research Question 4: Cumulative features underperform the other feature spaces

When we compare feature spaces across profiles and sampling methods cumulative features underperform the other feature spaces. However, the comparative performance of constant and term frequency features is dependent on the sampling method. Constant features outperform the other feature spaces when using random sampling, and term frequency features outperform when using exploitative sampling.

Figure 11(a) compares constant, cumulative, and term frequency features when sampling random seed documents and reordering with the query-specific profile. When comparing feature spaces over all iterations, we see that constant features consistently outperform, term frequency features clearly underperform,



(a) Random seed document sampling and reordering with the query-specific profile. MAP scores for iteration 1 to 20. Scores are presented for all feature spaces.



(b) Exploitative sampling and reordering with the global profile. MAP scores for iteration 1 to 20. Scores are presented for all feature spaces.

Fig. 11. Comparing all feature spaces for random document seed sampling and exploitative sampling.

and cumulative features perform second best. We see the same pattern in performance when randomly sampling and reordering with the global profile.

However, when using exploitative sampling, term frequency features consistently outperform the other feature spaces for both profiles. Figure 11(b) compares constant, cumulative, and term frequency features when using exploitative sampling from the seed documents and reordering with the global profile. When comparing feature spaces over all iterations for this method variation, term frequency features consistently outperform and constant features are a distant second best, while cumulative features perform worst for the vast majority of iterations. In combination with their poor performance when sampling randomly from seed documents, we conclude that, regardless of sampling method and profile type, cumulative features underperform.

Performance in feature space is correlated with sampling method The previous results suggest that performance in certain feature spaces—at least as measure by MAP and NDCG score—depends on the sampling strategy. Table 8 presents an overview of the correlations we have observed between sampling strategy and feature space. We expected that exploitative sampling would harm

Table 8. Approximate coupling between the relative performance of feature spaces compared across sampling strategies. The number of + signs indicates the number of feature spaces the current feature space performs better than.

Sampling \ Features	Cumulative	Constant	Term frequency
Exploitative	−	−	++
Random seeds	+	++	−

performance because it biases the collection of judged documents towards those ranked higher and gives us an incomplete picture of our feature space. The function we learn to rank with uses the squared hinge-loss and we can therefore decompose our error into:

$$\text{Error} = \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}.$$

For this decomposition, increasing the bias will increase the error [28, 34].

This explains the performance of the cumulative and constant features, but leaves the performance of term frequency features unexplained. The term frequency feature space functions quite differently from the retrieval score feature spaces. Most importantly, for the term frequency feature space to be useful the content—the actual email text—of judged documents must be helpful for reordering, whereas content is irrelevant for the retrieval score feature spaces.

Given this, we hypothesize that the content of highly ranked judged documents is more useful for reordering than the content of random judged documents. Because our retrieval method uses a language model, which has *already* ordered documents based on the relevance of their terms, the vector space formed from terms in highly ranked documents should contain more informative terms than the vector space formed from random documents. These more informative terms will likely be prominent enough to provide good coverage of the corpus and infrequent enough to distinguish relevant and non-relevant documents. As we have described them, these terms are similar to the terms which would appear in a *parsimonious* language model [37] of the corpus. They will therefore be more useful features when reordering documents.

Table 9 presents the top five terms from the set of relevant judged documents in iteration 2 for query 200 when using exploitative and random sampling. We

Table 9. Top 5 terms from the set of relevant judged documents on iteration 2 for query 200 when using exploitative and random sampling. We calculate term frequency within all relevant judged documents for query 200.

Sampling Strategy						Sum
Exploitative terms	`make`	`lots`	`Paragraph`	`Contract`	`Flair`	
Frequency	72	55	33	30	25	**215**
Random seed terms	`around`	`Chad`	`Gronvold`	`Nashville`	`Heather`	
Frequency	41	8	4	4	2	59

also show the frequency of these terms in the set of all relevant judged documents for query 200, and the sum over these frequencies. The sum of the frequencies is 215 for documents exploitative sampling, and 59 for documents sampled randomly. As expected, the terms in highly ranked documents—exploitative sampling—have a total frequency which is significantly higher than that of terms from randomly ranked documents.

When randomly sampling, three of the top five terms are very infrequent, they appear less than five times in all relevant documents. It is very likely that these three terms will make poor features in learning to rank. This evidence supports our hypothesis that highly ranked documents contain terms that are more discriminative within the corpus and will likely make better features when we apply learning to rank.

All features improve performance over time Based on Figures 11(a) and 11(b) we can make the further observation that scores increase over time for all feature spaces, although the amount by which they increase is dependent on the combination of feature space and sampling strategy. In the cumulative feature space, after we stop adding features from previous retrieval runs, we expect a plateau. In Figure 11(b) we see that, for the global profile and exploitative sampling, the cumulative feature space appears to plateau. For the beginning

iterations the standard deviation is $\sigma = 0.016$ (from iteration 2 to 8), much higher than the standard deviation of $\sigma = 0.0011$ for later iterations (from iteration 9 to 20). This provides quantitative evidence of a plateau.

A hybrid of retrieval score and term frequency features underperforms When we examined the per query scores for term frequency features, we noticed that for some queries reordering significantly underperforms. To address this we tested a feature space which combines term frequency features with the retrieval scores of each document as given by query expansion. Figure 12 compares MAP scores per query when reordering with term frequency features to reordering with term frequency features and retrieval scores.

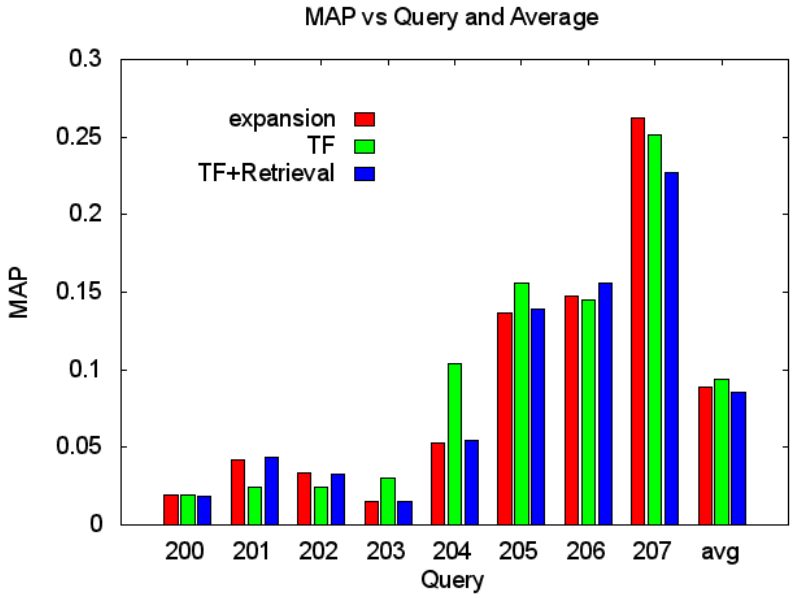


Fig. 12. Query profile, random seed documents, term frequency features and term frequency features with retrieval scores; MAP scores per query and averaged.

For queries 203 and 204, term frequency features outperform query expansion by 50% and 49% respectively. However, for queries 201 and 202, term frequency features underperform query expansion by 42% and 29% respectively. After adding retrieval scores as a feature in learning to rank we no longer underperform on queries 201 and 202. In fact we outperform query expansion by 6% for query 201 and only underperform by 4% for query 202. Unfortunately, gains on these queries are offset by losses for query 203 and 204, where we now improve performance by only 6% and 2% respectively. Overall, using retrieval

scores as additional features decreases scores by 3% when compared to query expansion, and 10% when using solely term frequency features.

This experiment served as a way to determine whether a hybrid method can improve scores and is worthy of future investigation. Through the per query improvements we see that a hybrid method can improve performance under certain conditions. This motivates future work constructing more in depth hybrid feature spaces. We expect that a hybrid feature space which combines term frequency features with constant features will perform well.

A query expansion oracle outperforms constant features When using the constant feature space we can consider learning to rank as a method of automatically choosing the number of query expansion terms to use.[23] We have shown that learning to rank can improve scores over those achieved when using only query expansion. We now investigate whether there is still more room to improve by comparing our scores to those of an oracle.

Figure 13 compares the best and worst scoring query expansion runs to our method when using constant features, the global profile, and randomly sampling 20 relevant and non-relevant seed documents. The maximum scores are those achievable if we always chose the number of query expansion terms which produced the highest scores, and the minimum scores are those if we always chose the number of query expansion terms which produced the lowest scores. We present MAP scores per query and averaged over all queries.

We see that our method is able to consistently outperform the minimum query expansion run but does not match the maximum query expansion run. This shows that although our method is able to choose amongst the query expansion runs such that it improves scores, it is not able to reach the maximum score. If we were to achieve the maximum query expansion scores we could potentially improve or method by 12% when averaged over all queries. However there are only a small number of judged documents and it is not clear that we can more effectively learn from them.

6.5 Kendall's τ is positively correlated with MAP score

It is possible that we can use the relationship between the amount of reordering, which is indicated by Kendall's τ, and performance to create methods that automatically determine when learning to rank is appropriate to apply and which judged documents are outliers. A positive correlation implies less reordering is better for performance, a negative correlation the opposite, and no correlation suggests the amount of reordering and performance are unrelated. Figures 14(a) and 14(b) present scatter plots of Kendall's τ and MAP score for each iteration and query, as well as the Pearson correlation coefficient ρ (r in the title of each subplot), which quantifies the linear dependence between τ and MAP score.

[23] More precisely, it is a method of automatically choosing a weighting over the influence of multiple retrieval orderings with varying numbers of query expansion terms when creating a final document ordering.

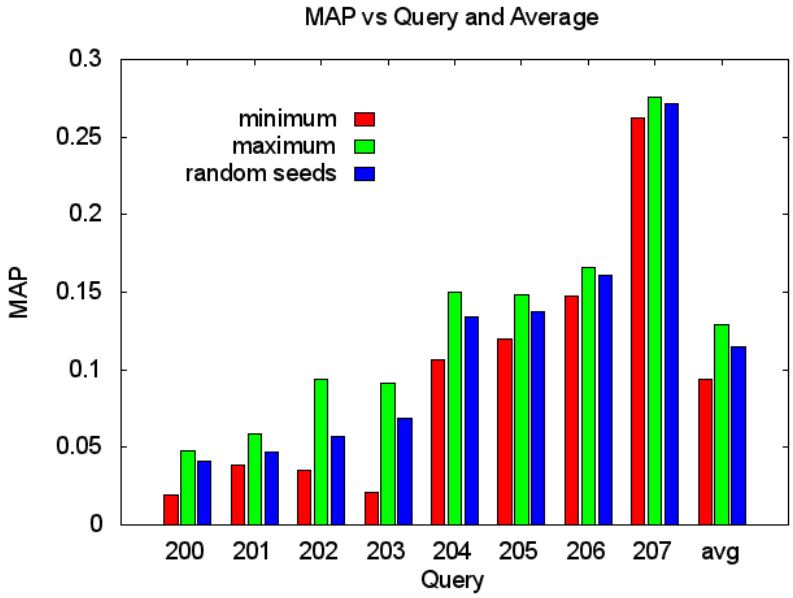
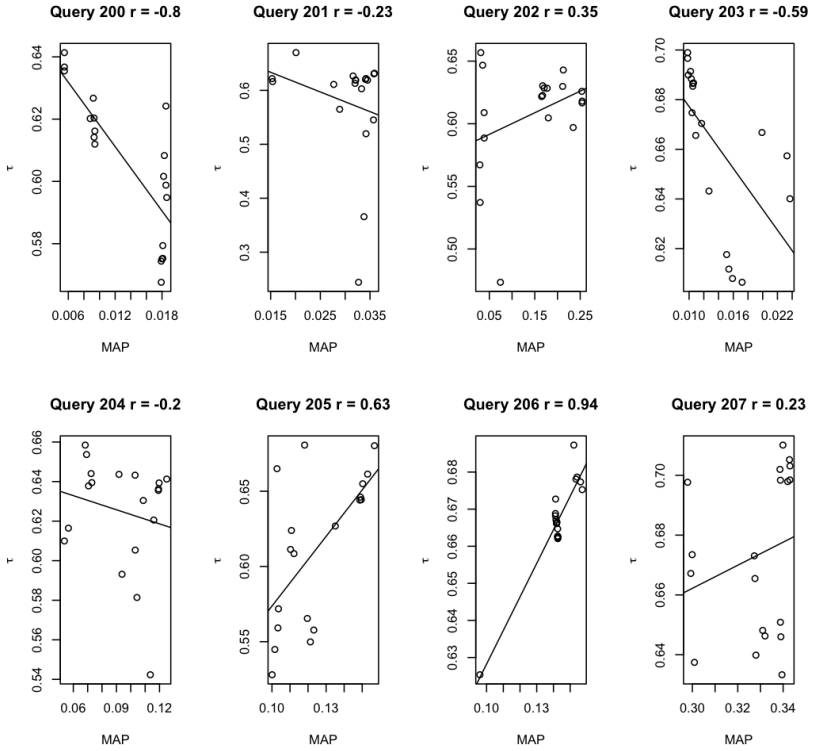


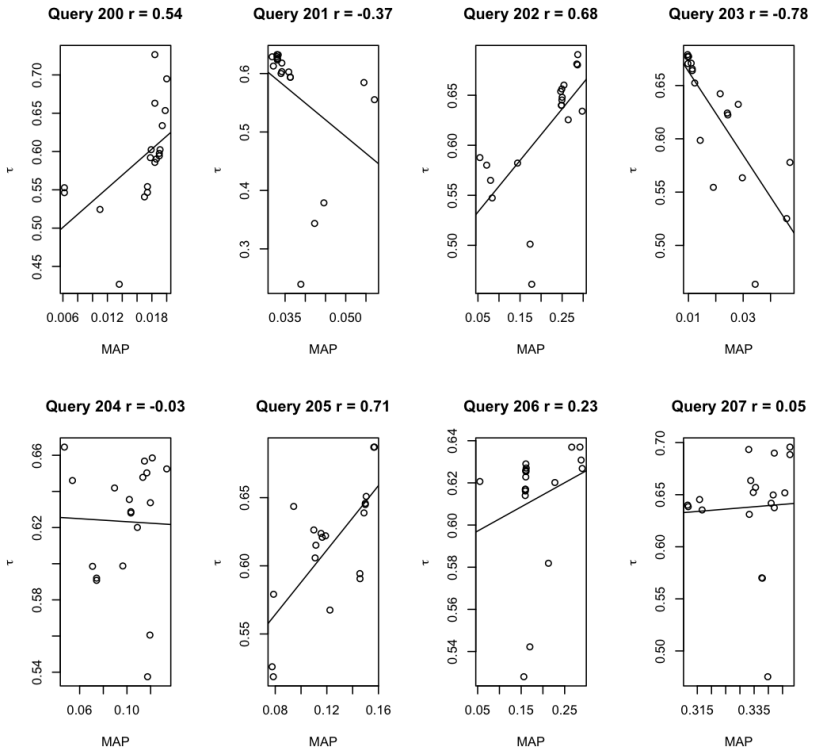
Fig. 13. Learning to rank with the query profile, random seed documents, and constant features compared to the maximum and minimum query expansion run scores. MAP scores are per query and averaged.

If we follow accepted practice [21] and label queries with $|\rho| \leq 0.3$ as uncorrelated, with $\rho > 0.3$ as positively correlated, and $\rho < -0.3$ as negatively correlated; there are 6 uncorrelated, 6 positively correlated, and 4 negatively correlated queries. This provides reason to suspect that Kendall's τ may be positively correlated with MAP score, and thus the amount of reordering may be negatively correlated with MAP score. As we have seen in the context of search diversification during previous work [55], these results appear to show that putting more trust in the original ordering is an effective strategy for re-ranking.

When we compare the query-specific profile plots in Figure 14(a) to the global profile plots in Figure 14(b) we see that the per query distribution of Kendall's τ and MAP score pairs are somewhat similar. In fact, per query the sign of ρ is the same between the two profiles for all queries except query 200. Queries 201 through 207 were taken from the TREC 2009 Legal interactive task, in which participants chose the documents that relevance judgements were made for. Query 200 is unique in that it was new for TREC 2010 Legal and the organizers chose the documents for which relevance judgements were made. The difference which we see in the sign of ρ could be related to the fact that different procedures were used to select the judged documents for query 200 as compared to queries 201 through 207.



(a) Global profile, random seed documents, constant features; linear regression of MAP and Kendall's τ for all iterations per query.



(b) Query profile, random seed documents, constant features; linear regression of MAP and Kendall's τ for all iterations per query.

Fig. 14.

We can also analyze the plots in Figures 14(a) and 14(b) to determine which documents are outliers. For example, in query 202's plot in Figure 14(a) there is an iteration with a low MAP score and a very low Kendall's τ, shown in the bottom left. It is likely that a document added during this iteration is an outlier.

6.6 Selecting terms by frequency is robust

An essential part of effectively using relevance feedback for query expansion is to select the correct terms with which to expand the original query. Although term frequency is an effective and efficient manner of selecting terms, it does not necessarily perform best, and is certainly not the only term selection method. In additional experiments we used term frequency (TF) inverse-document-frequency (IDF)—together referred to as (TFIDF)—to build a vector space model, then used k-Means clustering to group relevant documents according to their features in this model.[24] Next we chose the top $n = 10$ terms from these clusters, and then chose the top m terms closest to their respective cluster's center. In preliminary tests we found that using these terms for query expansion showed a slight improvement over TF when expanding with 5 terms and a larger improvement when expanding with 20 terms.

Unfortunately the improvements shown in preliminary experiments for query expansion did not translate into improvements in our iterative method. This poor performance is likely because k-Means term selection is ineffective when there are too few instances. At the simplest level, with 1 instance this method reduces to TFIDF, suggesting that k-Means term selection is inappropriate for our iterative system. We conclude that TF is a more robust method than k-Means term selection and that this is a reason to prefer it to more complex methods. None the less, it would be worthwhile to experiment with other methods of extracting representative terms from the corpus to use in our search system, e.g. [6, 7].

[24] We use Apache Mahout to build the TFIDF vector space model and to cluster, `http://mahout.apache.org/`

7 Conclusion

Current search systems primarily focus on ad-hoc search settings in which interactions are very limited. In information retrieval, and especially in recall oriented search, we can significantly improve the order of search results by learning from user's preferences, which are indicated through their interactions with the search system. When users describe their preferences by spending the time to judge documents we should make their effort worthwhile by best using the information they provide to their benefit.

In e-discovery search tasks there are often an overwhelming number of potentially relevant documents along with time and financial constraints which affect the search process. Search for e-discovery has not yet been studied in depth, although its popularity has significantly increased because a large and growing number of court cases involve searching through documents found during e-discovery. Because of the high cost attached to document judgement it is essential that search system built for e-discovery use judged documents to improve the result ordering as much as possible.

We address these search challenges by making use of judged documents through a novel method which exploits the information they contain on two levels: in learning to rank and in query expansion. Furthermore, our method uses the information in judged documents to improve results immediately as the user provides feedback. These methods help our search system to offset the cost incurred by judging documents.

We evaluate our method using an e-discovery search task taken from TREC Legal. Our combined learning to rank and query expansion method improves search result ordering, as measured by MAP and NDCG, beyond that which is achievable when using only query expansion. The email documents that make up the corpus we evaluate our method on are not significantly different from the documents in other corpuses (e.g. web pages), which indicates that our method's improvement in scores may be a general result that will apply to other corpuses.

We avoid the pitfalls of negative relevance feedback [98–100] by ignoring non-relevant documents during query expansion and then later using these documents as negative instances when learning to rank. We experiment with a global profile which combines query-specific judged documents with judged documents for different queries. The global profile is designed to capture terms which are useful when learning to rank for any search within the same context or general information need.

In summary, our search system first uses standard language model based retrieval to rank documents for the user's query. The user then judges documents, which are made available to our system. The system then builds an expanded query, in the cumulative feature space; a set of expanded queries, in the constant feature space; or a vector space model from the judged documents' terms, in the term frequency feature space. With the information provided by the judged documents and their feature vectors, the system learns a model and reorders documents to better align them with judgements made by the user. Depending on how document judgements are requested both the constant and term fre-

quency feature spaces are capable of improving document ordering beyond that achievable when using query expansion alone.

7.1 Lessons learned

We find that the global profile underperforms, more documents lead to improved performance, and sampling from informative (seed) documents is best but the most effective strategy depends on the feature space. We further find that constant features outperform cumulative and term frequency features for random sampling, while term frequency features outperform constant and cumulative features for exploitative sampling. Below we review each of these conclusions and mention additional conclusions.

The global profile does not improve performance in most cases Our first research question asks whether a global or query-specific profile performs best. We suggested that by using a broader collection of documents the global profile may be able to address over-fitting and reduce the negative effects of unrepresentative judged documents. However, we find that adding instances judged for other queries and not the current query reduces the focus of our language model and reduces scores. In the majority of cases the query-specific profile outperforms the global profile, although the global profile can outperform the query-specific profile for certain queries and variations of our method.

We find that for retrieval score feature spaces the global profile does not improve performance and it is better to choose documents for learning that are specific to the task and context at hand, in the case of information retrieval this is the current query. In these feature spaces the negative effects of including documents in the global profile that may disagree with the current query's judgements outweighs the benefits of a broader profile. However, when using for term frequency features and exploitative sampling, the global profile outperforms the query-specific profile. These results suggest that there are global patterns, but the current method of combining judged documents does not make optimal use of these patterns. Future work can be done to develop new methods of combining judged documents, which may result in higher performance.

Increase in performance dependent on sampling strategy Our second research question investigates the effects of learning to rank as we increase the number of judged documents. As expected, we find the general trend is that more judged documents improve our ability to learn, although from iteration to iteration a new judged document may decrease performance. A profile with more judged documents gives us a more representative set of documents for use in learning to rank.

We find that the method of sampling judged documents significantly affects the performance over time. There is less improvement over time when randomly sampling all documents, but significant improvement when sampling from seed documents. This is because the seed documents are chosen to be informative.

The benefit of adding a judged document that is expected to be particularly informative is greater than that of adding a judged document that is not especially informative.

Because our feature spaces are directly (term frequency features) or indirectly (constant and cumulative features) dependent on the contents of our documents (their terms) a larger profile of judged documents leads to a more representative feature space. We conclude that for all our tested feature spaces and sampling methods having more judged documents leads to improved performance. A larger number of judged documents is particularly effective when we request judgements for documents that we expect to be relevant.

The appropriate sampling strategy depends on the feature space Our third research question compares various strategies of selecting judged documents. The first strategy samples from the highest ranked document, as in pseudo-relevance or implicit feedback; the second strategy samples from a set of informative documents; and the third strategy samples from all documents. We find that sampling randomly from the seed documents outperforms other sampling strategies for constant and cumulative features, while exploitative sampling performs best for term frequency features.

Choosing a sampling strategy is analogous to choosing a method with which to request document judgements from the user. We conclude that requesting judgements randomly is better for retrieval score features and requesting judgements for the highest ranked document is better for term frequency features. We find that, for all feature spaces, it is better to request judgements for documents we expect to be informative than to request judgements at random.

Feature space and sampling strategy effectiveness are correlated Our fourth and final research question compares using different feature spaces for learning to rank. We find that learning from constant features outperforms the other feature spaces when sampling randomly and term frequency features outperform when using exploitative sampling. Exploitative sampling harms performance because it biases the collection of judged documents towards those ranked higher and gives us an incomplete picture of our feature space. When using retrieval score features increasing the bias will increase the error.

When using the term frequency feature space the actual text of judged documents must be helpful for reordering, not only retrieval scores. Highly ranked judged documents have more prominent terms and are therefore more useful for reordering when compared to random judged documents. We use this to explain why the exploitative sampling strategy outperforms other sampling strategies when using term frequency features.

The cumulative feature space is never the best performing feature space. This is because the cumulative feature space does not take advantage of newly judged documents. Although using different sets of judged documents may reduce the negative effects of a harmful judged document, not using the full set of judged

documents outweighs these benefits. We conclude that the cumulative feature space should not be used in practice.

We evaluate a hybrid feature space that contains term frequency features and retrieval score features, but find that it does not outperform either individual feature space. The hybrid feature space achieves its goal of improving performance on queries for which term frequency features perform worse than query expansion. However hybrid features perform worse than term frequency features on queries for which term frequency features had previously significantly outperformed the baseline query expansion scores. When these two factors are combined the hybrid feature space in unable to outperform the term frequency feature space.

Less reordering may be correlated with improved performance We analyzed the Kendall's τ rank correlation coefficient and found that for some queries the global profile performs less reordering. This is because the global profile contains more documents and therefore a new document is less influential. In cases were the newly judged document negatively affects scores through its influence in the query-specific profile, it has less influence in the global profile and using the global profile can improve scores.

We found inconclusive evidence that the amount of reordering is negatively correlated with performance. This corroborates previous results in which a conservative reordering method—one that only reorders when there is a significant chance this will improve performance—is better at improving performance than a liberal method, which does more reordering.

Term frequency is a robust term selection method When selecting terms for query expansion and for the term frequency feature space we chose terms with the highest frequency. We experimented with an additional term selection algorithm based on k-Means and did not find any gains in performance, likely because of this algorithm's weaknesses when used with small data sets. High frequency terms are known to be representative of document text and we find that by choosing them we are able to select terms which are useful in both query expansion and as features in a term frequency feature space.

7.2 Future work

Because our method is not idiosyncratic to the dataset we used, we expect that our results and the conclusions drawn above will be applicable to many information retrieval corpora. One approach would be to use implicit judgements based on click data from query logs in place of explicit document judgements. We believe our method will be especially useful for recall oriented retrieval tasks, such as patent, academic, and medical search. By experimenting with other corpora, future work could evaluate our method on larger sets of judged documents and gain a better understanding of how our method depends on the size of the dataset.

There is work that can be done to optimize parameters. Future work could optimize the number of terms used in the original query expansion, the number of terms used in expansion when building constant features for learning to rank, as well as the number of constant features to use. Another area for future work is to explore additional hybrid feature spaces and optimize the manner in which we combine features. Additionally, future work could experiment with alternative term selection methods, such as TFIDF. This will be combined with work examining the term distribution to help determine the best method for term selection and the number of terms to use in the term frequency feature space.

Predicting performance with Kendall's τ In future work we plan to further investigate the correlation between higher Kendall's τ and better performance when learning to rank. Suppose we have reordered a document list using learning to rank and $score'$ is the score after reordering while $score$ is the score before reordering. If there exists a function:

$$f(\tau) = \Pr(score' > score),$$

we can use the amount of reordering, as measured by τ, to predict the probability that the score of the reordered list is greater than that of the original list. From our observations, this probability will increase with τ. Given this we can then selectively return the reordered list or the original list based on the probability that it will increase performance.

Alternative approaches to building preference pairs Equation 9 in Section 3.3 shows that we build preference pairs by taking positively judged documents over negatively judged ones. Future work could form an additional preference pair set under the assumption that positively judged documents are preferred to not judged documents, which are in turn preferred to negatively judged documents:

$$\{(i,j) \in \mathcal{P} | (\langle \mathbf{d}_i, 1\rangle \in P \wedge \langle \mathbf{d}_j, 1\rangle \notin P) \vee (\langle \mathbf{d}_i, -1\rangle \notin P \wedge \langle \mathbf{d}_j, -1\rangle \in P)\},$$

and variations of this. In situations where we have few judged documents this may improve performance.

Exploiting graphs and structure The Enron corpus consists of structured emails, which provide us with a graph based upon the `to` and `from` fields. This graph could be used to produce a PageRank [69] or other spectral rankings [39, 97]. We could further adapt the query-specific and global profiles to build a personalized [33, 46] or topic based PageRank [35] for each query.

The emails' structure also includes a `subject` field, footer, inline document quotations, and implicit salutations. There is a growing area of research which addresses using structure in documents [47, 50, 109] and queries [29, 110] for improving retrieval performance. Considering that the text from the recipient

field was polluting our expansion terms (we discussed the prevalence of the term `Corp`), we expect significant benefits from developing a retrieval system that considers document structure.

Topic modeling for term selection and feature space construction It is sometimes unclear how topic modeling can be applied to information retrieval. We show that using a method similar to topic modeling improves retrieval performance. The feature analysis we perform functions similarly to the low dimensional document representation found by topic modeling [4, 38, 101, 104, 111]. In future work we plan to explore this analogy and exploit topic models to discover low dimensional document representations. Future work could use topic detection and tracking [76, 80] to group documents for feature space construction and weight documents according to their importance within the collection [32].

Incremental learning applied to visualization for personalization Future work could apply our method to building incrementally updated relevant term visualizations by extending work that uses keywords and geographic space analogies [2, 15, 105]. This work could then compare visualizations of the terms in highly ranked documents and other ralsionships between documents [25] before and after reordering to gain further insight into the effects of our various feature spaces and document profiles. We also expect that this type of visualization will assist users in browsing search results.

Applications to the TREC Legal Interactive Task In the TREC Legal Interactive Task the search system and team participants can request additional document judgements from a researcher assigned to each query, the "topic authority." In future work we would like to combine our method with active learning to choose documents which are likely to be informative. Our method could then be applied to the TREC Legal Interactive Task.

Bibliography

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM '09*, pages 5–14, 2009.

[2] J.-w. A. Ahn and P. Brusilovsky. Adaptive visualization of search results: Bringing user models to visual analytics. *Information Visualization*, 8(3): 167–179, 2009.

[3] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *ECML*, volume 3201 of *Lecture Notes in Computer Science*, pages 39–50. Springer, 2004.

[4] G. Anthes. Topic models vs. unstructured data. *Communications of the ACM*, 53(12):1618, 2010.

[5] M. Balabanovic. An interface for learning multi-topic user profiles from implicit feedback. In *Conference on Recommender Systems*, 1998.

[6] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 113–120, New York, NY, USA, 2006. ACM.

[7] D. M. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *JMLR*, 3: 993–1022, 2003.

[8] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *WSCD '09: Proceedings of the 2009 workshop on Web Search Click Data*, pages 56–63, New York, NY, USA, 2009. ACM.

[9] P. Boldi, F. Bonchi, C. Castillo, and S. Vigna. Query reformulation mining: models, patterns, and applications. *Inf. Retr.*, 14(3):257–289, 2011.

[10] C. Brandt, T. Joachims, Y. Yue, and J. Bank. Dynamic ranked retrieval. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 247–256, New York, NY, USA, 2011. ACM.

[11] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 25–32, New York, NY, USA, 2004. ACM.

[12] C. J. C. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical report, 2010.

[13] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

[14] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *NIPS*, pages 193–200, 2006.

[15] G. Cai. Geovibe: A visual interface for geographic digital libraries. In *Visual Interfaces to Digital Libraries [JCDL 2002 Workshop]*, pages 171–187, London, UK, 2002. Springer-Verlag.

[16] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98*, pages 335–336, 1998.

[17] O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with svms. *Information Retrieval*, 13(3):201–215, 2010.

[18] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[19] P. A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 7–14, New York, NY, USA, 2007. ACM.

[20] C. Clarke, M. Kolla, G. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR '08*, pages 659–666, 2008.

[21] J. Cohen. *Statistical power analysis for the behavioral sciences*. L. Erlbaum Associates, 1988. URL `http://books.google.com/books?id=TlON2lRAO9oC`.

[22] J. G. Conrad. E-discovery revisited: the need for artificial intelligence beyond information retrieval. *Artif. Intell. Law*, 18:321–345, December 2010.

[23] G. Cormack, M. R. Grossmand, B. Hedin, and D. W. Oard. Overview of the trec 2010 legal track, 2010.

[24] A. Culotta, A. Liu, M. Cordover, B. Borden, and S. Strickland. IT-Discovery at TREC 2010 Legal. In E. M. Voorhees and L. P. Buckland, editors, *The Nineteenth Text REtrieval Conference Proceedings (TREC 2010)*. National Institute of Standards and Technology (NIST) the Defense Advanced Research Projects Agency (DARPA) and the Advanced Research and Development Activity (ARDA), Nov. 2010.

[25] F. Das-Neves, E. A. Fox, and X. Yu. Connecting topics in document collections with stepping stones and pathways. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 91–98, New York, NY, USA, 2005. ACM.

[26] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *ICML*, pages 208–215, 2008.

[27] E. D'hondt, S. Verberne, N. Oostdijk, and L. Boves. Re-ranking based on Syntactic Dependencies in Prior-Art Retrieval. In *Proceedings of the Dutch-Belgium Information Retrieval workshop 2010 (DIR 2010)*, 2010.

[28] P. Domingos. A unified bias-variance decomposition for zero-one and squared loss. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 564–569. AAAI Press, 2000.

[29] S. Dominich. *The Modern Algebra of Information Retrieval*, volume 24 of *The Information Retrieval Series*. Springer, Berlin, 2008.

[30] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, December 2003.

[31] A. Garron and A. Kontostathis. Applying Latent Semantic Indexing on the TREC 2010 Legal Dataset. In E. M. Voorhees and L. P. Buckland, editors, *The Nineteenth Text REtrieval Conference Proceedings (TREC*

2010). National Institute of Standards and Technology (NIST) the Defense Advanced Research Projects Agency (DARPA) and the Advanced Research and Development Activity (ARDA), Nov. 2010.

[32] S. Gerrish and D. Blei. A language-based approach to measuring scholarly impact. In *ICML*, pages 375–382, 2010.

[33] D. Gleich and M. Polito. Approximating personalized pagerank with minimal use of web graph data. *Internet Mathematics*, 3(3), 2007.

[34] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[35] T. H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. Technical Report 2003-29, Stanford InfoLab, 2003. Extended version of the WWW2002 paper on Topic-Sensitive PageRank.

[36] B. Hedin, S. Tomlinson, J. R. Baron, and D. W. Oard. Overview of the trec 2009 legal track, 2009.

[37] D. Hiemstra, S. Robertson, and H. Zaragoza. Parsimonious language models for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 178–185, New York, NY, USA, 2004. ACM.

[38] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[39] V. Hollink and M. van Someren. Optimal link categorization for minimal retrieval effort. In *Proceedings of Sixth Dutch-Belgian Information Retrieval Workshop*, pages 65–72, Delft, The Netherlands, 2006.

[40] V. Hollink, T. Tsikrika, and A. de Vries. Semantic vs term-based query modification analysis. In *Proceedings of the tenth Dutch-Belgian Information Retrieval Workshop*, 2010.

[41] F. Hopfgartner and J. Jose. Semantic user profiling techniques for personalised multimedia recommendation. *Multimedia Systems*, 16:255–274, 2010.

[42] F. Hopfgartner and J. M. Jose. Semantic user profiling techniques for personalised multimedia recommendation. *Multimedia Syst.*, 16(4-5):255–274, 2010.

[43] T. Joachims. Making large-scale svm learning practical. LS8-Report 24, Universität Dortmund, LS VIII-Report, 1998.

[44] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[45] T. Joachims. Training linear svms in linear time. In *KDD*, pages 217–226, 2006.

[46] S. Kamvar. *Numerical Algorithms for Personalized Search in Self-organizing Information Networks*. Princeton University Press, 2010.

[47] G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum. The yago-naga approach to knowledge discovery. *SIGMOD Rec.*, 37:41–47, March 2009.

[48] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[49] O. Kurland, L. Lee, and C. Domshlak. Better than the real thing?: iterative pseudo-query processing using cluster-based language models. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 19–26, New York, NY, USA, 2005. ACM.

[50] K. Lerman, R. Ghosh, and J. H. Kang. Centrality metric for dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, MLG '10, pages 70–77, New York, NY, USA, 2010. ACM.

[51] X.-S. Li, S. Li, W.-R. Xu, G. Chen, and J. Guo. Weakly supervised relevance feedback based on an improved language model. In *Natural Language Processing and Knowledge Engineering (NLP-KE), 2010 International Conference on*, pages 1–5, 2010.

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

[53] Y. Liu, T.-Y. Liu, B. Gao, Z. Ma, and H. Li. A framework to compute page importance based on user behaviors. *Inf. Retr.*, 13:22–45, February 2010.

[54] C. M. Lorenzetti and A. G. Maguitman. A semi-supervised incremental algorithm to automatically formulate topical queries. *Inf. Sci.*, 179:1881–1892, May 2009.

[55] P. Lubell-Doughtie and K. Hofmann. Improving Result Diversity using Probabilistic Latent Semantic Indexing. In *Proceedings of the Dutch-Belgium Information Retrieval workshop 2011 (DIR 2011)*, 2011.

[56] C. Lundquist, D. A. Grossman, and O. Frieder. Improving relevance feedback in the vector space model. In *Proceedings of the sixth international conference on Information and knowledge management*, CIKM '97, pages 16–23, New York, NY, USA, 1997. ACM.

[57] M. Lupu, J. Huang, J. Zhu, and J. Tait. Trec-chem: large scale chemical information retrieval evaluation at trec. *SIGIR Forum*, 43:63–70, December 2009.

[58] J. Luxenburger. *Modeling and Exploiting User Search Behavior for Information Retrieval*. PhD thesis, Universität des Saarlandes, December 2008.

[59] J. Luxenburger, S. Elbassuoni, and G. Weikum. Task-aware search personalization. In *SIGIR*, pages 721–722, 2008.

[60] J. Luxenburger, S. Elbassuoni, and G. Weikum. Matching task profiles and user needs in personalized web search. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 689–698, New York, NY, USA, 2008. ACM.

[61] Y. Lv and C. Zhai. Adaptive relevance feedback in information retrieval. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 255–264, New York, NY, USA, 2009. ACM.

[62] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[63] E. Margolis. Surfin' safari - why competent lawyers should research on the web. *Yale Journal of Law and Technology*, 10, 2007.

[64] N. Matthijs and F. Radlinski. Personalizing web search using long term browsing history. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 25–34, New York, NY, USA, 2011. ACM.

[65] E. Meij. *Combining concepts and language models for information access.* PhD thesis, Amsterdam, Netherlands, 2010.

[66] E. Meij. Combining concepts and language models for information access. *SIGIR Forum*, 45:80–80, May 2011.

[67] K. Nigam, A. Mccallum, and T. M. Mitchell. *Semi-Supervised Text Classification Using EM*, chapter 3. MIT Press, Boston, 2006.

[68] M. Okabe and S. Yamada. Semisupervised query expansion with minimal feedback. *IEEE Trans. on Knowl. and Data Eng.*, 19:1585–1589, November 2007.

[69] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[70] G. Pandey and J. Luxenburger. Exploiting session context for information retrieval - a comparative study. In *ECIR*, pages 652–657, 2008.

[71] D. P. Papadopoulus, V. S. Kalogeiton, and A. Arampatzis. DUTH does Probabilities of Relevance at the Legal Track. In E. M. Voorhees and L. P. Buckland, editors, *The Nineteenth Text REtrieval Conference Proceedings (TREC 2010)*. National Institute of Standards and Technology (NIST) the Defense Advanced Research Projects Agency (DARPA) and the Advanced Research and Development Activity (ARDA), Nov. 2010.

[72] P. Pirolli. *Information Foraging Theory: Adaptive Interaction with information.* Oxford University Press, New York, 2007.

[73] B. Piwowarski and M. Lalmas. A quantum-based model for interactive information retrieval. In *Proceedings of the 2nd International Conference on Theory of Information Retrieval: Advances in Information Retrieval Theory*, ICTIR '09, pages 224–231, Berlin, Heidelberg, 2009. Springer-Verlag.

[74] J. M. Ponte. A language modeling approach to information retrieval. Master's thesis, Amherst, MA, USA, 1998.

[75] J. M. Ponte. Language models for relevance feedback. In W. B. Croft, editor, *Advances in Information Retrieval*, volume 7 of *The Information Retrieval Series*, pages 73–95. Springer US, 2002.

[76] J. Qiu, L. Liao, and P. Li. News recommender system based on topic detection and tracking. In *RSKT '09*, pages 690–697, 2009.

[77] J.-M. Renders. Xerox participation to legal trec 2010 - learning task, 2010.

[78] S. Y. Rieh and H. Xie. Analysis of multiple query reformulations on the web: the interactive information retrieval context. *Inf. Process. Manage.*, 42(3):751–768, May 2006.

[79] J. J. Rocchio. Relevance feedback in information retrieval. pages 313–323. 1971.

[80] R. Rossi and J. Neville. Modeling the evolution of discussion topics and communication to improve relational classification. In *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, pages 89–97, New York, NY, USA, 2010. ACM.

[81] M. Sanderson. Ambiguous queries: test collections need more sense. In *SIGIR '08*, pages 499–506, 2008.

[82] R. L. T. Santos, C. Macdonald, and I. Ounis. On the suitability of diversity metrics for learning-to-rank for diversity. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Beijing, China, 2011. ACM.

[83] R. L. T. Santos, C. Macdonald, and I. Ounis. Intent-aware search result diversification. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Beijing, China, 2011. ACM.

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

[85] L. Sharp and M. Lange. Juggling the worlds of paper and electronic discovery. *ABTL Report*, 6, 2004.

[86] F. J. Smith and K. Devine. Storing and retrieving word phrases. *Inf. Process. Manage.*, 21(3):215–224, 1985.

[87] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: a language-model based search engine for complex queries. Technical report, in Proceedings of the International Conference on Intelligent Analysis, 2005.

[88] B. Tan. *A study of language models for exploiting user feedback in information retrieval.* PhD thesis, Champaign, IL, USA, 2009.

[89] S. Tomlinson. Learning Task Experiments in the TREC 2010 Legal Track. In E. M. Voorhees and L. P. Buckland, editors, *The Nineteenth Text REtrieval Conference Proceedings (TREC 2010).* National Institute of Standards and Technology (NIST) the Defense Advanced Research Projects Agency (DARPA) and the Advanced Research and Development Activity (ARDA), Nov. 2010.

[90] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, March 2002.

[91] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma. Frank: a ranking method with fidelity loss. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 383–390, New York, NY, USA, 2007. ACM.

[92] H. R. Turtle. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9:187–222, 1991.

[93] V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[94] S. Verberne, H. van Halteren, S. Raaijmakers, D. Theijssen, and L. Boves. Learning to Rank QA data. In *Proceedings of the Learning to Rank workshop at SIGIR 2009*, pages 41–48, 2009.

[95] S. Verberne, M. Hinne, M. van der Heijden, E. Hoenkamp, W. Kraaij, and T. P. van der Weide. How does the library searcher behave? a contrastive

study of library search against ad-hoc search. In M. Braschler, D. Harman, and E. Pianta, editors, *CLEF (Notebook Papers/LABs/Workshops)*, 2010.

[96] S. Verberne, H. Van Halteren, S. Raaijmakers, D. Theijssen, and L. Boves. Learning to Rank for Why-Question Answering. *Information Retrieval*, 2010.

[97] S. Vigna. Spectral ranking. *CoRR*, abs/0912.0238, 2009. informal publication.

[98] X. Wang. *Improving Web Search for Difficult Queries.* PhD thesis, Champaign, IL, USA, 2009.

[99] X. Wang, H. Fang, and C. X. Zhai. Improve retrieval accuracy for difficult queries using negative feedback. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 991–994. ACM New York, NY, USA, 2007.

[100] X. Wang, H. Fang, and C. Zhai. A study of methods for negative relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 219–226. ACM, 2008.

[101] Y. Wang, H. Li, H. Wang, and K. Q. Zhu. Toward topic search on the web. Technical report, Microsoft, 2011.

[102] M. J. Welch, J. Cho, and C. Olston. Search result diversity for informational queries. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 237–246, New York, NY, USA, 2011. ACM.

[103] D. Widdows. *Geometry and Meaning.* Center for the Study of Language and Inf, November 2004.

[104] S. Williamson, C. Wang, K. Heller, and D. Blei. The ibp compound dirichlet process and its application to focused topic modeling. In *ICML*, pages 1151–1158, 2010.

[105] M. L. Wilson, B. Kules, M. Schraefel, and B. Shneiderman. From keyword search to exploration: Designing future search interfaces for the web. *Foundations and Trends in Web Science*, 2:1–97, 2010.

[106] Z. Xu and R. Akella. A bayesian logistic regression model for active relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 227–234, New York, NY, USA, 2008. ACM.

[107] Z. Xu and R. Akella. Active relevance feedback for difficult queries. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 459–468, New York, NY, USA, 2008. ACM.

[108] Z. Yin, M. Shokouhi, and N. Craswell. Query expansion using external evidence. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 362–374, Berlin, Heidelberg, 2009. Springer-Verlag.

[109] L. Zhao and J. Callan. A generative retrieval model for structured documents. In *CIKM*, pages 1163–1172, 2008.

[110] L. Zhao and J. Callan. Effective and efficient structured retrieval. In *CIKM*, pages 1573–1576, 2009.

[111] J. Zhu, A. Ahmed, and E. P. Xing. Medlda: maximum margin supervised topic models for regression and classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1257–1264, New York, NY, USA, 2009. ACM.

[112] L. Zighelnic and O. Kurland. Query-drift prevention for robust query expansion. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 825–826, New York, NY, USA, 2008. ACM.

[113] G. K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley, 1949.