Dynamic Ranked Retrieval

Christina Brandt
Cornell University
Ithaca, NY, USA
ccb35@cornell.edu

Thorsten Joachims
Cornell University
Ithaca, NY, USA
tj@cs.cornell.edu

Yisong Yue
Carnegie Mellon Univ.
Pittsburgh, PA, USA
yisongyue@cmu.edu

Jacob Bank
Cornell University
Ithaca, NY, USA
jeb369@cornell.edu

ABSTRACT

We present a theoretically well-founded retrieval model for dynamically generating rankings based on interactive user feedback. Unlike conventional rankings that remain static after the query was issued, dynamic rankings allow and anticipate user activity, thus providing a way to combine the otherwise contradictory goals of result diversification and high recall. We develop a decision-theoretic framework to guide the design and evaluation of algorithms for this interactive retrieval setting. Furthermore, we propose two dynamic ranking algorithms, both of which are computationally efficient. We prove that these algorithms provide retrieval performance that is guaranteed to be at least as good as the optimal static ranking algorithm. In empirical evaluations, dynamic ranking shows substantial improvements in retrieval performance over conventional static rankings.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*relevance feedback, retrieval models*

General Terms

Algorithms, Theory

Keywords

Diversified Retrieval, Relevance Feedback, Decision Theory

1. INTRODUCTION

With the dominance of short, one- or two-word queries in many retrieval settings, most queries are ambiguous at some level. For such ambiguous queries, there is often no single ranking that satisfies all users and query intents. While result diversification aims to provide a "compromise ranking" that provides some utility for all intents, diversification necessarily sacrifices recall and there is a danger that a diversified ranking will not adequately satisfy the information needs of any of the users.

WSDM'11, February 9–12, 2011, Hong Kong, China.

In this paper, we propose a retrieval model that goes beyond the conventional single-ranking approach. We define a decision-theoretic model that naturally combines diversity with high recall on all query intents. The key idea is to make the ranking "dynamic" – namely, allowing it to change in response to user interactions after the query was issued.

From the user's perspective, this may look as illustrated in Figure 1. This interface is inspired by and adapted from the SurfCanyon.com search engine [9], but other result layouts are also possible. In this example, the user first receives a conventional diversified ranking in response to the query "SVM" (Figure 1, left). However, by clicking or mousing over a result that matches the user's intent, additional indented results are inserted into the original ranking[1] (Figure 1, middle). This process can be repeated multiple levels deep (Figure 1, right). We argue that this interaction is natural, since the process resembles navigating a drop-down menu and since users are already familiar with result indentation. In the example, the indented results have greatly improved recall for the user's information need on the learning method "Support Vector Machine". While there is only a single relevant document in the original ranking, the final ranking covers many aspects of the learning method.

From the system's perspective, the task is no longer the prediction of a single ranking, but instead of a tree of results. This "dynamic ranking tree" (see Section 3) describes how users can interact with the results (i.e. expand results), and what indented rankings they will see in response. Instead of predicting a single ranking, the retrieval system can optimize the ranking tree so that users with various intents will find many relevant results given their intent-specific interactions. Intuitively, the system must construct the ranking tree so that diversity in the top-level ranking provides appropriate leads for all intents, while the lower levels provide additional relevant results and allow for further refinement.

This paper makes the following contributions towards this interactive retrieval setting, which we call "Dynamic Ranked Retrieval". First, we propose a concise decision-theoretic model for reasoning about and evaluating Dynamic Ranked Retrieval systems that can generalize conventional measures such as nDCG to the interactive setting. Optimizing expected retrieval performance in this model naturally leads to a well-founded trade-off between diversity and recall, providing a sound theoretical basis for developing retrieval algorithms. Second, we present two such algorithms for predicting ranking trees. Both algorithms are efficient, and we

[1]Alternatively, one could show additional results to the right of the original ranking in a multi-column layout.

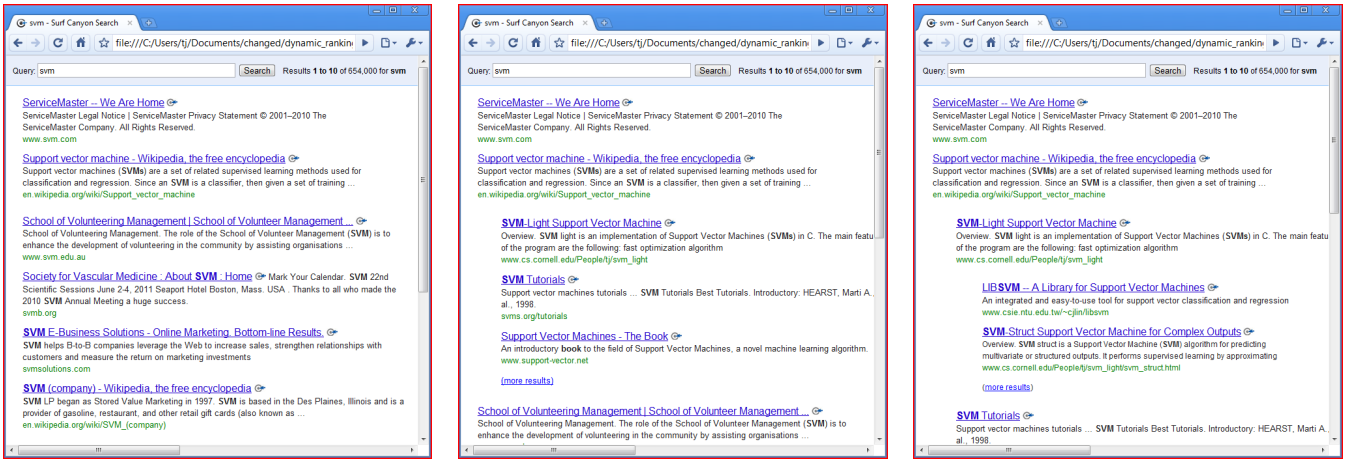

Figure 1: Example of user interacting with dynamic ranking.

prove that they always perform at least as well as conventional static ranking systems for a large class of performance measures. Third, we empirically show that these Dynamic Ranked Retrieval algorithms can provide substantial gains in retrieval quality for ambiguous queries compared to the best static ranking (e.g. improving Prec@10 by 15-20 percentage points on TREC diversity tasks). Fourth, since Dynamic Ranked Retrieval requires knowledge about dependencies between multiple documents, we show that generalized models of these dependencies can be learned and integrated into our algorithms.

2. RELATED WORK

Our work lies at the intersection of two research directions, diversified retrieval and relevance feedback, with an aim towards modeling interactive retrieval settings. Both research directions seek to address the issue of query ambiguity, i.e. when the information need or intent is unclear given the query. Prior work on diversified retrieval and prior work on relevance feedback have been largely complementary (and disjoint). This paper demonstrates that the modeling of interactive retrieval settings can benefit substantially from considering the two aspects simultaneously.

Diversified Retrieval. Result diversification is commonly used to provide good coverage for a given query [19]. The prediction goal is typically stated (with varying degrees of formality) as minimizing the worst case scenario of a user finding no or few relevant documents among the top results. This can be naturally approached by retrieving a set of results that optimizes for some compromise between (estimated) relevance and novelty [7, 28]. Developing suitable evaluation measures remains an active and growing research direction (cf. [28, 27, 8, 2]).

More recently, researchers have tackled diversified retrieval by viewing it as a coverage problem. Here, the goal is to directly optimize the number of users "covered" (e.g. users who find at least one relevant result) [27, 20, 2, 11]. When a gold standard measure of coverage is unavailable at test time, one can instead use a proxy coverage measure or model (possibly learned from a labeled training set) based on readily available features of the candidate documents [27, 11].

Relevance Feedback. In the relevance feedback setting, feedback from users or expert judges are used to augment the query in order to generate more informative models of the information need [23]. At a high level, there are three types of relevance feedback: explicit feedback from users or expert judges, implicit feedback collected from users as they interact with the system, and pseudo feedback, which is collected without any human response. Such feedback is used to build a more refined model of user intent. Well-known methods include vector space model approaches [22, 24], language model approaches [18, 29], and query expansion approaches [4, 10, 5]. These methods all follow the same general motivation: the resulting query intent should be close with respect to a chosen similarity measure to the documents provided by relevance feedback.

Interactive Retrieval. Broadly speaking, interactive retrieval refers to any retrieval setting where the system and its users exchange multiple rounds of interaction per query or session. Since user interactions by definition provide feedback to the retrieval system, relevance feedback can be viewed as a special case of interactive retrieval. At a high level, user interactions can be categorized along two dimensions: explicit versus implicit feedback, and system-driven versus user-driven interactions.

System-driven interactions involve the retrieval system directly asking users to provide (explicit) feedback in order to better understand the information need. Examples include explicitly asking users to select the most relevant documents from a presented pool [25, 26], or asking users which facets (or subtopics) best capture their query intent [30]. While intrusive to the user experience, such approaches can be effective when dealing with difficult queries [26].

User-driven interactions involve settings where the retrieval system makes recommendations that users can accept or ignore, or creates an interface for users to provide more feedback if the users choose to do so. Typically, the retrieval system responds to a sequence of queries within a particular session with recommendations, such as additional results or query reformulation suggestions [15, 3]. Users provide implicit feedback when they click on results or accept a suggested query reformulation. Such feedback can then be used to provide more useful recommendations when responding to subsequent queries in the session [6].

In this space of related work, our goal is to develop a general decision-theoretic framework for reasoning about entire

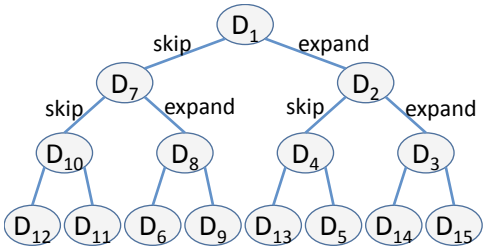

Figure 2: Illustration of a dynamic ranking tree.

sequences of interactions, similar to [12]. We focus on a simple user-driven interaction setting which naturally combines diversity and relevance feedback, can accommodate both explicit and implicit feedback, and is well suited for modeling existing interactive retrieval systems such as [9]. Our work also has an affinity to models for faceted search [16], but does not assume any meta-data about facets. Finally, our user interaction model is similar to Incremental Relevance Feedback [1] in that each relevance feedback signal immediately and incrementally affects the next result to display.

3. DYNAMIC RANKED RETRIEVAL

We now formalize the goal of Dynamic Ranked Retrieval into a well-founded yet simple decision-theoretic model. The core component is the notion of a ranking tree, which replaces the static ranking of a conventional retrieval system. An example is shown in Figure 2. The nodes in the tree correspond to individual results (i.e. documents), and each user's search experience corresponds to a path in the tree. The path a particular user takes depends on that user's actions, in particular whether the user decides to expand a result to view the corresponding indented ranking. Expanding a result corresponds to taking the right branch of the corresponding node in the ranking tree, and skipping corresponds to taking the left branch. Mapping the ranking tree in Figure 2 to the example in Figure 1, the user skipped over D_1 = "ServiceMaster", expanded D_7 = "SVM Wikipedia", and then expanded D_8 = "SVM-light".

For simplicity, we first consider the setting where users act according to the following deterministic interaction policy, which we call π_{det}: users always expand (go right) at nodes representing relevant documents and skip (go left) non-relevant documents. Note that users with different query intents consider different documents as relevant, and so will take different paths through the tree. We will explore other user policies later, in particular policies involving noisy user behavior.

It is now very natural to score the retrieval quality of a particular user's search experience via the documents encountered on her path through the ranking tree. Note that the traversed path corresponds to the final dynamic ranking presented in the user's browser, so that the i-th document on the traversed path corresponds to the i-th document the user sees. Thus, the traversed path is essentially a user-specific ranking, which we can evaluate using existing performance measures like nDCG, average precision, or Precision@k.

More formally, let $P(r|q)$ denote the distribution of information needs r for query q. Each information need r (i.e. relevance profile) is a vector of relevance judgments, with one judgment for each document in the collection. This models the fact that different users issuing the query q will consider different sets of documents as relevant. Let Ψ be a ranking tree for query q. Then a user's relevance profile r and interaction policy π determine her path σ through Ψ. For π_{det} defined above, this path is deterministic. For noisy user behavior where the probability of expanding a result d is no longer 0 or 1, a user's experience will be sampled from a distribution of paths, $P(\sigma|\Psi, r, \pi)$, where π is now a non-deterministic interaction policy. We assume that all users behave according to a common interaction policy π (e.g. expand relevant results w.p. 75%), and differ only in which documents they find relevant.

Table 1: Example of rel. profile distribution $P(R|q)$.

	$P(r_i)$	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12} ...
r_1	0.2	1	1	1	0	0	0	0	0	0	0	0	0 ...
r_2	0.2	1	0	0	1	1	0	0	0	0	0	0	0 ...
r_3	0.2	0	0	0	0	0	1	1	0	0	0	0	0 ...
r_4	0.2	0	0	0	0	0	0	1	1	1	0	0	0 ...
r_5	0.2	0	0	0	0	0	0	0	0	0	1	1	0 ...

We can now apply standard retrieval measures and define the expected retrieval performance for a particular query q and ranking tree Ψ as

$$U(\Psi|q,\pi) = \sum_r \sum_\sigma U(\sigma|r) P(\sigma|\Psi, r, \pi) P(r|q). \quad (1)$$

The overall quality of the retrieval algorithm A is then simply the expectation over the distribution of queries,

$$U(A|\pi) = \sum_q U(A(q)|q,\pi) P(q). \quad (2)$$

The utility $U(\sigma|r)$ of a path σ given a relevance profile r can be measured by any conventional ranked retrieval measure. In particular, one can define the following dynamic extensions of Precision at k (Prec@k), average precision (AP@k), discounted cumulative gain (DCG@k), and normalized discounted cumulative gain (nDCG@k) [17]. Let σ_i denote the i-th document on the user's path, $|r|$ denote the number of relevant documents in relevance profile r, and $\delta(\cdot)$ be a binary indicator function. Then these standard retrieval measures can be expressed as utility functions of the form:

$$\text{Prec@k: } U(\sigma|r) = \frac{1}{k} \sum_{i=1}^{k} r_{\sigma_i} \quad (3)$$

$$\text{AP@k: } U(\sigma|r) = \frac{1}{\min\{k, |r|\}} \sum_{i=1}^{k} \frac{\sum_{j=1}^{i} r_{\sigma_j}}{i} \delta(r_{\sigma_j}=1) \quad (4)$$

$$\text{DCG@k: } U(\sigma|r) = \sum_{i=1}^{k} \frac{r_{\sigma_i}}{\log_2(i+1)} \quad (5)$$

$$\text{nDCG@k: } U(\sigma|r) = \frac{1}{DCG^*(r)} \left(\sum_{i=1}^{k} \frac{r_{\sigma_i}}{\log_2(i+1)} \right) \quad (6)$$

The normalization constant $DCG^*(r)$ for nDCG corresponds to the utility of the best possible ranking for the given relevance profile. Note that an estimate of $U(A|\pi)$ can easily be computed by sampling q, r and σ as is typically done for static ranked retrieval.

Conventional static ranked retrieval corresponds to the special case where users always choose the left branch and never expand results. Note that for this user policy, our

Algorithm 1 StaticMyopic for Static Ranking

Init: $\text{StaticMyopic}(q, \mathcal{D}) = \text{StaticMyopic}(q, \mathcal{D}, ())$
Recurse: $\text{StaticMyopic}(q, \mathcal{D}, \sigma)$
  $d = \text{argmax}_{d \in \mathcal{D}} \left\{ \sum_r (U(\sigma \oplus d|r) - U(\sigma|r)) P(r|q) \right\}$
  $n = \text{StaticMyopic}(q, \mathcal{D} \setminus \{d\},\ \sigma \oplus d)$
return$(list(d, n))$

Algorithm 2 DynamicMyopic for Dynamic Ranking

Init: $\text{DynamicMyopic}(q, \mathcal{D}, \pi)$
  $= \text{DynamicMyopic}(q, \mathcal{D}, \pi, (), ())$
Recurse: $\text{DynamicMyopic}(q, \mathcal{D}, \pi, \sigma, \gamma)$
  $d = \text{argmax}_{d \in \mathcal{D}} \left\{ \sum_r (U(\sigma \oplus d|r) - U(\sigma|r)) P(r|q, \pi, C_\sigma = \gamma) \right\}$
  $l = \text{DynamicMyopic}(q,\ \mathcal{D} \setminus \{d\},\ \pi,\ \sigma \oplus d,\ \gamma \oplus [0])$
  $r = \text{DynamicMyopic}(q,\ \mathcal{D} \setminus \{d\},\ \pi,\ \sigma \oplus d,\ \gamma \oplus [1])$
return(tree(d,l,r))

evaluation model in Eq. (2) reduces to the intent-aware evaluation measures proposed in [2]. For this special case of rankings σ, we will use the abbreviated notation $U(\sigma|q)$ to refer to the expected utility defined in Eq. (1). In addition, if the query is unambiguous and there is only a single relevance profile r (i.e. $P(r|q) = 1$), then our evaluation measures reduce to the conventional definitions of Prec@k, AP@k, DCG@k, and nDCG@k.

To illustrate the dynamic ranking model, consider again the ranking tree example Ψ shown in Figure 2 that was computed for some fixed query q. Table 1 shows the relevance profiles of five query intents for query q. Assuming those intents have uniform probability $P(r_i|q) = 1/5$ and users follow policy π_{det}, a user with relevance profile r_1 will follow the path $(d_1, d_2, d_3, ...)$ and will encounter all three relevant documents in the first three positions. This leads to a DCG@4 of 2.13. Similarly, users with profile r_2 will follow the path $(d_1, d_2, d_4, d_5, ...)$, where only d_2 is not relevant, resulting in a DCG@4 of 1.93. Users with profiles r_3, r_4, and r_5 will experience DCG@4 values of 1.06, 1.56, and 0.93, respectively. Taking the expectation over all profiles, the dynamic DCG@4 of the ranking tree Ψ is $DCG@4(\Psi|q, \pi_{det}) = 1.52$. Note that the best possible static ranking σ can only achieve $DCG@4(\sigma|q) = 0.84$ for this distribution of relevance profiles.

4. ALGORITHMS

When query ambiguity is ignored, it is well known that sorting by probability of relevance (or by expected marginal utility) produces the best possible ranking for virtually all evaluation measures proposed to date [21]. This is known as the "probability ranking principle". But for measures that account for query ambiguity, such as the measures defined above and their special cases proposed in [2], the situation is less clear. In particular, how can one compute a ranking tree, or even a static ranking, to achieve high expected performance over the distribution of intents for a given query? Furthermore, for practical settings, the dynamic ranking algorithms should not be substantially more computationally expensive than conventional ranking algorithms. In the following, we first consider the case of static ranking, and then propose two efficient algorithms for computing dynamic ranking trees for which we give performance guarantees relative to the optimal static ranking.

4.1 Algorithm for Static Ranking

Before presenting the algorithms for constructing dynamic ranking trees, we first consider how to construct a static ranking with high expected performance under the measures discussed above. Algorithm 1, which we call StaticMyopic, can be seen as the straightforward extension of the probability ranking principle to the case of ambiguous queries. We will show in Section 5 that StaticMyopic produces optimal static rankings for DCG, nDCG, and Prec@k, but, surprisingly, is not optimal for AP.

Algorithm 1 is a greedy method that iteratively appends (denoted by $\oplus$) the document with the highest expected gain in utility,

$$d = \underset{d \in \mathcal{D}}{\text{argmax}} \left\{ \sum_r (U(\sigma \oplus d|r) - U(\sigma|r)) P(r|q) \right\} \quad (7)$$
$$= \underset{d \in \mathcal{D}}{\text{argmax}} \left\{ \sum_r U(\sigma \oplus d|r) P(r|q) \right\}.$$

We call the quantity inside the curly brackets of Eq. (7) the *marginal utility* $U(d|q, \sigma)$ of d given σ. Solving the argmax requires only a single pass through the candidate set $\mathcal{D}$.[2] Summing over the relevance profiles is inexpensive if the number of profiles with non-zero probability is small, as it is for the TREC datasets used for our experiments.

Note that for measures like DCG and Prec@k, the utility gain $U(\sigma \oplus d|r) - U(\sigma|r)$ depends only on the relevance r_d of d and the current length of σ, but not on the full relevance profile r or the specific documents in σ. We call such functions *modular* (see Section 5 for a precise definition). Given a modular performance measure, the marginal utility in Eq. (7) can be rewritten as

$$U(d|q, \sigma) = f(|\sigma|) \sum_{r_d} r_d P(r_d|q),$$

where $P(r_d|q)$ is the (marginal) probability of relevance of document d, and $f(|\sigma|)$ depends only on the length of σ (see Table 2). Thus the argmax can be simplified to

$$d = \underset{d \in \mathcal{D}}{\text{argmax}} \left\{ \sum_{r_d} r_d P(r_d|q) \right\}.$$

It is now easy to see that, for modular performance measures, StaticMyopic is equivalent to sorting the documents by expected utility as in conventional retrieval systems. The same simplification applies to $nDCG@k$ after scaling each relevance profile by $1/DCG^*(r)$. However, this simplification does not apply to $AP@k$.

4.2 Algorithms for Dynamic Ranking

In the following, we propose two efficient algorithms for constructing dynamic ranking trees. Both algorithms build ranking trees top-down by recursively adding child nodes to the current leaves (similar to most decision-tree learning algorithms). Unlike StaticMyopic, document selection is performed by conditioning on the sequence of user interactions (e.g. result expansions and skips) that led the user to that node.

[2]In practice, one would generate a reasonably-sized candidate set (a few hundred documents) using a conventional retrieval function as a filter.

Algorithm 3 DynamicLookahead for Dynamic Ranking

```
1: Init: DynamicLookahead(q, D, π)
         = DynamicLookahead(q, D, π, (), ())
2: Recurse: DynamicLookahead(q, D, π, σ, γ)
3:     d = argmax_{d∈D} {
4:         Σ_r (U(σ ⊕ d|r) − U(σ|r)) P(r|q, π, C_σ = γ)
5:         +P(C_d=0|q, π, C_σ=γ) Û_l + P(C_d=1|q, π, C_σ=γ) Û_r
6:     }
7:     l = DynamicLookahead(q, D \ {d}, π, σ ⊕ d, γ ⊕ [0])
8:     r = DynamicLookahead(q, D \ {d}, π, σ ⊕ d, γ ⊕ [1])
9: return(tree(d,l,r))
```

4.2.1 Dynamic Myopic Algorithm

The first algorithm, called DynamicMyopic (see Algorithm 2), can be thought of as the natural extension of StaticMyopic. When creating a new leaf node, DynamicMyopic keeps track of the path σ leading to this node, as well as the sequence $\gamma \in \{0,1\}^*$ of skips/expands required to navigate there. It then selects a document that most improves the utility of the updated path,

$$d = \operatorname*{argmax}_{d\in\mathcal{D}} \left\{ \sum_r (U(\sigma \oplus d|r) - U(\sigma|r)) P(r|q,\pi,C_\sigma=\gamma) \right\}$$
$$= \operatorname*{argmax}_{d\in\mathcal{D}} \left\{ \sum_r U(\sigma \oplus d|r) \frac{P(C_\sigma=\gamma|q,r,\pi)P(r|q,\pi)}{P(C_\sigma=\gamma|q,\pi)} \right\}$$
$$= \operatorname*{argmax}_{d\in\mathcal{D}} \left\{ \sum_r U(\sigma \oplus d|r) P(C_\sigma=\gamma|q,r,\pi)P(r|q) \right\}.$$

We use C_σ to denote the random variable describing the skips/expands for the documents in σ. Therefore, $P(C_\sigma = \gamma|q,r,\pi)$ is the probability that a user with relevance profile r and interaction policy π will visit the current node in the tree. The second equality above follows from two applications of the law of conditional probabilities, and the last equality follows from observing that the denominator is constant and that the distribution of relevance profiles is assumed to not depend on the user interaction policy (i.e. $P(r|q,\pi) = P(r|q)$). Similar to StaticMyopic, solving the argmax requires only one pass through the documents in the candidate set $\mathcal{D}$.

Note that it is not necessary to explicitly generate the entire ranking tree for each query or user session. In particular, it suffices to generate solely the branches that are currently displayed to the user. In other words, solving the argmax in Line 2 of Algorithm 2 can be done lazily – it is only explicitly required after the user session has traversed to that particular node in the ranking tree.

Analogous to StaticMyopic, the argmax for Prec@k, DCG, and nDCG@k can be rewritten using only the marginal distributions $P(r_d|q,\pi,C_\sigma=\gamma)$ of $d \in \mathcal{D}$.

$$d = \operatorname*{argmax}_{d\in\mathcal{D}} \left\{ \sum_{r_d} r_d P(r_d|q,\pi,C_\sigma=\gamma) \right\}$$

In the learning experiments in Section 7, we will directly estimate the relevance distribution $P(r_d|q,\pi,C_\sigma=\gamma)$ using a linear model, which is very efficient to evaluate.

4.2.2 Dynamic Lookahead Algorithm

The second algorithm, called DynamicLookahead (see Algorithm 3), uses a lookahead estimate when determining which document to select as a new child node. Rather than considering only a document's immediate marginal utility, the algorithm chooses the document d which maximizes the marginal utility plus the (approximate) utilities $\hat{U}_l$ and $\hat{U}_r$ of its two subtrees. We approximate $\hat{U}_l$ and $\hat{U}_r$ using StaticMyopic, which we will show to provide a lower bound on the utility of each subtree.

Note that the complexity of solving the argmax of DynamicLookahead for $Prec@k$, $DCG@k$, and $nDCG@k$, is $O(|\mathcal{D}|^2 \log|\mathcal{D}|)$, compared to $O(|\mathcal{D}|)$ for DynamicMyopic. We will empirically evaluate the benefit of this computational expense.

Table 2: Instantiating Prec@k, DCG, and nDCG as modular functions according to Eq. (8).

	Prec@k	DCG	nDCG
$f(i)$	$\delta(i \le k)$	$1/\log(i+1)$	$1/\log(i+1)$
$U(d,r)$	r_d	r_d	$r_d/DCG^*(r)$

5. THEORETICAL ANALYSIS

We first introduce a notion called modularity[3] which will be useful for proving our theoretical results.

Definition 1. *We call a utility function U **modular** if, for any static ranking σ, it satisfies the following form:*

$$U(\sigma|r) = \sum_{i=1}^{k} f(i) U(\sigma_i|r), \tag{8}$$

where $f(i)$ is non-negative and monotonically decreasing function, and $U(\sigma_i|r)$ is a non-negative utility function of a single document that does not depend on rank position.

Linear combinations of modular functions are also modular, so the expected utility of a static ranking algorithm A_{static}, $U(A_{static})$ from Eq. (2), is modular if the per-query expected utility, $U(A_{static}|q)$ from Eq. (1), is modular.

It is relatively straightforward to see that instantiating Eq. (2) with Prec@k, DCG@k, or nDCG@k results in a modular utility function (see Table 2). Note that although Eq. (2) is defined for dynamic ranking algorithms, the modularity property applies only with respect to static rankings.

5.1 Analyzing Static Rankings

As discussed in Section 4.1, for modular utility functions, StaticMyopic reduces to sorting by expected utility of the individual documents. As such, Theorem 1 below follows immediately from Definition 1.

Theorem 1. StaticMyopic *and the Probability Ranking Principle both produce optimal static rankings for modular utility functions.*

However, it can be shown that both StaticMyopic and sorting by expected utility (i.e. the probability ranking principle) can be suboptimal for AP when the query is ambiguous.

Observation 1. *Both* StaticMyopic *and the Probability Ranking Principle can produce suboptimal static rankings for average precision and AP@k.*

[3]Our definition of modularity is a special case of the more general definition of modularity for set-based utility functions (cf. [13]).

Consider the following counterexample. Consider a candidate set $\mathcal{D}$ with three documents for a query q with two relevance profiles $r_1 = (1, 0, 0)$ and $r_2 = (0, 1, 1)$. Let $P(r_1|q) = 1/3$ and $P(r_2|q) = 2/3$. Then for position one of the static ranking, all three documents achieve the maximum marginal utility w.r.t. AP@k (with $k \geq 3$) of 1/3. However, the optimal static rankings are $(2, 3, 1)$ and $(3, 2, 1)$ with an AP@k of 0.78, while the best possible ranking that starts with document 1 (e.g. $(1, 2, 3)$) has an AP@k of only 0.72.

In the above counterexample, if one modifies the probabilities of the two relevance profiles to be uniformly 1/2, then applying the Probability Ranking Principle by sorting by expected relevance can also be suboptimal.

5.2 Analyzing Dynamic Rankings

We now investigate the performance gain that dynamic ranking trees can achieve over static rankings. We first define the performance criterion, which we call *adaptivity gain*.

DEFINITION 2. *For a specified utility function U and user interaction policy π, the **adaptivity gain** of (dynamic) ranking algorithm A is*

$$U(A|\pi) - U(\text{StaticMyopic}).$$

We will show in the following that both DynamicMyopic and DynamicLookahead always achieve non-negative adaptivity gain, i.e. they will never perform worse than StaticMyopic.

We first introduce some notation. Specifically, we need notation to describe the behavior of our ranking algorithms when conditioned on a *pre-existing context* (σ, γ), which consists of a list of already presented documents σ and their click information γ. Such context, for instance, can be generated by first running some other retrieval algorithm. Most notably, the per-query utility function in Eq. (1) now becomes a conditional utility measure.

In the following notation, we will suppress the query q for brevity. Given a pre-existing context (σ, γ), we can write the conditional utility of a ranking tree Ψ as

$$U(\Psi|\pi, \sigma, \gamma) = \left(\sum_r \sum_{\sigma'} U(\sigma \oplus \sigma'|r) P(\sigma'|\Psi, r, \pi, C_\sigma = \gamma) P(r|C_\sigma = \gamma, \pi) \right) - U(\sigma|r),$$

which reduces to the unconditional utility defined in Eq. (1) in the case where the pre-existing context (σ, γ) is empty. We only consider algorithms that never show repeat documents, i.e. the σ' in the summation above is always disjoint from σ. For modular utility functions, we can write the conditional utility of a static ranking σ' as

$$U(\sigma'|\pi, \sigma, \gamma) = \sum_{i=1}^{k-|\sigma|} U(\sigma'_i|\pi, \sigma, \gamma), \tag{9}$$

where

$$U(d|\pi, \sigma, \gamma) = f(i + |\sigma|) \sum_r U(d|r) P(r|C_\sigma = \gamma, \pi).$$

We can thus also consider adaptivity gain for conditional utilities by computing

$$U(A|\pi, \sigma, \gamma) - U(\text{StaticMyopic}|\pi, \sigma, \gamma).$$

We will also use the shorthand notations of SM, DM and DL for StaticMyopic, DynamicMyopic and DynamicLookahead, respectively.

We now prove the following useful lemma, which compares the utility of StaticMyopic with the utility of a ranking algorithm that performs one iteration of DynamicMyopic followed by StaticMyopic for the remainder (i.e. a ranking tree where the root node has two subrankings).

LEMMA 1. *For any modular utility function U, any pre-existing context (σ, γ), and any known user interaction policy π, we have*

$$U(\text{SM}|\pi, \sigma, \gamma) \leq U(d^*|\pi, \sigma, \gamma) + \sum_{\gamma' \in \{0,1\}} U(\text{SM}|\pi, \sigma \oplus d^*, \gamma \oplus \gamma') P(\gamma'|\pi, C_\sigma = \gamma).$$

where $d^* = \text{argmax}_{d \in \mathcal{D} \setminus \sigma} U(d|\pi, \sigma, \gamma)$.

PROOF. We can write $U(\text{SM}|\pi, \sigma, \gamma)$ as

$$\begin{aligned} U(\text{SM}|\pi, \sigma, \gamma) &= \max_{\sigma'} U(\sigma'|\pi, \sigma, \gamma) \\ &= U(d^*|\pi, \sigma, \gamma) + \max_{\sigma'} U(\sigma'|\pi, \sigma \oplus d^*, \gamma), \end{aligned} \tag{10}$$

where Eq. (10) follows from the definition of modular functions. We can further write the max term in Eq. (10) as

$$\begin{aligned} &\max_{\sigma'} U(\sigma'|\pi, \sigma \oplus d^*, \gamma) \\ &\quad = \max_{\sigma'} \sum_{\gamma' \in \{0,1\}} U(\sigma'|\pi, \sigma \oplus d^*, \gamma \oplus \gamma') P(\gamma'|\pi, C_\sigma = \gamma) \\ &\quad \leq \sum_{\gamma' \in \{0,1\}} \max_{\sigma'} U(\sigma'|\pi, \sigma \oplus d^*, \gamma \oplus \gamma') P(\gamma'|\pi, C_\sigma = \gamma) \\ &\quad = \sum_{\gamma' \in \{0,1\}} U(\text{SM}|\pi, \sigma \oplus d^*, \gamma \oplus \gamma') P(\gamma'|\pi, C_\sigma = \gamma), \end{aligned}$$

which completes the proof. □

THEOREM 2. *For modular utility function U, DynamicMyopic has a non-negative adaptivity gain for any pre-existing context (σ, γ) and user interaction policy π.*

PROOF. We prove this by induction on the remaining interaction length k. The base case $k = 1$ is trivial. In the inductive case, we assume for interaction length $k - 1$ that DynamicMyopic has non-negative adaptivity gain for any pre-existing context and user interaction policy.

Using Lemma 1 and its definition of d^*, we can write the conditional utility of StaticMyopic, $U(\text{SM}|\pi, \sigma, \gamma)$, as bounded by

$$\begin{aligned} &\leq U(d^*|\pi, \sigma, \gamma) + \sum_{\gamma' \in \{0,1\}} U(\text{SM}|\pi, \sigma \oplus d^*, \gamma \oplus \gamma') P(\gamma'|\pi, C_\sigma = \gamma) \\ &\leq U(d^*|\pi, \sigma, \gamma) + \sum_{\gamma' \in \{0,1\}} U(\text{DM}|\pi, \sigma \oplus d^*, \gamma \oplus \gamma') P(\gamma'|\pi, C_\sigma = \gamma) \\ &= U(\text{DM}|\pi, \sigma, \gamma), \end{aligned}$$

where the last inequality follows from the inductive hypothesis, and the last equality follows from observing that d^* is the document that is selected by DynamicMyopic. □

THEOREM 3. *For modular utility function U, DynamicLookahead has a non-negative adaptivity gain for any pre-existing context (σ, γ) and user interaction policy π.*

PROOF. We prove this by induction on the remaining interaction length k. The base case $k = 1$ is trivial. In the inductive case, we assume for interaction length $k-1$ that DYNAMICLOOKAHEAD has non-negative adaptivity gain for any pre-existing context and user interaction policy.

Using Lemma 1 and its definition of d^*, we can write the conditional utility of STATICMYOPIC, $U(\text{SM}|\pi,\sigma,\gamma)$, as bounded by

$$\leq U(d^*|\pi,\sigma,\gamma) + \sum_{\gamma' \in \{0,1\}} U(\text{SM}|\pi, \sigma \oplus d^*, \gamma \oplus \gamma') P(\gamma'|\pi, C_\sigma = \gamma)$$

$$\leq \max_{d \in \mathcal{D} \setminus \sigma} \Bigg\{ U(d|\pi,\sigma,\gamma) + \sum_{\gamma' \in \{0,1\}} U(\text{SM}|\pi, \sigma \oplus d, \gamma \oplus \gamma') P(\gamma'|\pi, C_\sigma = \gamma) \Bigg\}. \quad (11)$$

Note that Eq. (11) is exactly the criterion optimized by DYNAMICLOOKAHEAD in Algorithm 3. Let d^{**} denote the maximizer of Eq. (11). We can write Eq. (11) as

$$= U(d^{**}|\pi,\sigma,\gamma) + \sum_{\gamma' \in \{0,1\}} U(\text{SM}|\pi, \sigma \oplus d^{**}, \gamma \oplus \gamma') P(\gamma'|C_\sigma = \gamma, \pi)$$

$$\leq U(d^{**}|\pi,\sigma,\gamma) + \sum_{\gamma' \in \{0,1\}} U(\text{DL}|\pi, \sigma \oplus d^{**}, \gamma \oplus \gamma') P(\gamma'|C_\sigma = \gamma, \pi)$$

$$= U(\text{DL}|\pi,\sigma,\gamma),$$

where the last inequality follows from the inductive hypothesis, and the last equality follows from observing that d^{**} is the document that is selected by DYNAMICLOOKAHEAD. □

The two theorems characterize the performance of the dynamic algorithms w.r.t. the optimal static ranking. Comparing to the optimal static ranking seems to be a promising direction also for future theoretical results, since constructing trees with good approximation factor to the optimal trees is known to be computationally intractable [13]. However, further restrictions to the problem setting may allow proving stronger performance guarantees w.r.t. both baselines.

6. EVALUATING THE ADAPTIVITY GAIN

The previous section showed that the dynamic rankings computed by DYNAMICMYOPIC and DYNAMICLOOKAHEAD are always at least as good as the static ranking computed by STATICMYOPIC, i.e. the adaptivity gain (Definition 2) is never negative. But is there actually a substantial benefit? We explore this question empirically on two datasets: the TREC 6-8 Interactive Track, which we will refer to as INTERACTIVE, and the Diversity Task of the TREC 18 Web Track on the Clueweb collection, which we will refer to as WEB.

Both datasets are accompanied by queries with relevance judgments for different subtopics. For each query, we consider each of its subtopics to be the relevance profile of a different information need. The relevance judgments are binary. On the INTERACTIVE dataset, the number of profiles in each of the 20 topics ranges from 7 to 56, with an average of 20. On average, a profile in the INTERACTIVE dataset is associated with 3 relevant documents. On the WEB dataset, the number of profiles in each of the 50 topics ranges from 2 to 8 with a mean of 4.75, and the average number of documents relevant to a profile is 33. Unlike in the TREC evaluation on WEB, we do not discard relevance profiles where the relevance judges did not find any relevant documents. On average, there are 0.75 such empty profiles per query. We follow [8] and assume that both the distribution of queries (topics) $P(q)$ and the distribution of profiles (subtopics) for each query $P(R = r|q)$ is uniform. For INTERACTIVE, no information about the distribution of topics and profiles is given, so we assume that $P(q)$ is uniform and $P(R = r|q)$ is proportional to the number of relevant documents in r.

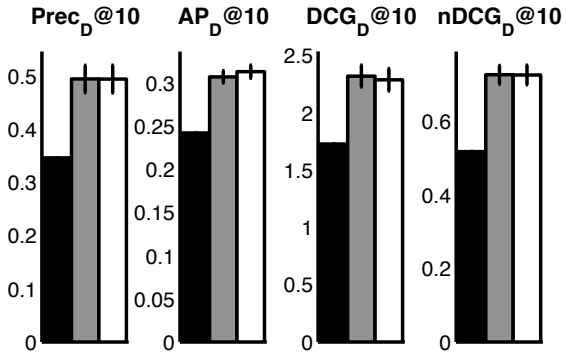

Figure 3: Adaptivity Gain on the INTERACTIVE dataset for DYNAMICMYOPIC (gray) and DYNAMICLOOKAHEAD (light) with respect to STATICMYOPIC (black). The errorbars show the standard error of the respective adaptivity gain.

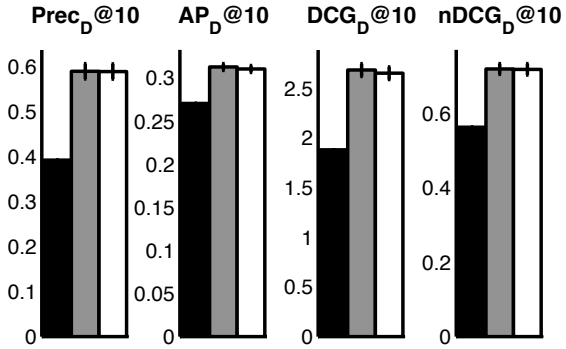

Figure 4: Same as Figure 3, but for the WEB dataset.

An implementation of our algorithms and experiment setup is available at `http://dynamicranking.joachims.org`.

6.1 Experiment 1: Average Adaptivity Gain

For the first experiment, we eliminate all sources of uncertainty to most directly compare the retrieval quality of STATICMYOPIC, DYNAMICMYOPIC, and DYNAMICLOOKAHEAD. In particular, all algorithms have perfect knowledge of the relevance profiles and their distribution $P(R = r|q)$, and users behave deterministically according to π_{det} (i.e. they always expand relevant documents in their relevance profile, and never expand irrelevant documents). We measure performance using the dynamic variants of Prec@10, AP@10, DCG@10, and nDCG@10 as defined in Section 3, which reduce to the intent-aware metrics of Agrawal et al. [2] for static ranking. When evaluating using each dynamic ranking measure, we used that same measure as the utility function for constructing the dynamic ranking tree.

Figure 3 and Figure 4 show average retrieval performance for the INTERACTIVE and the WEB datasets, respectively. For all datasets and evaluation measures, the dynamic ranking methods substantially outperform the static ranking.

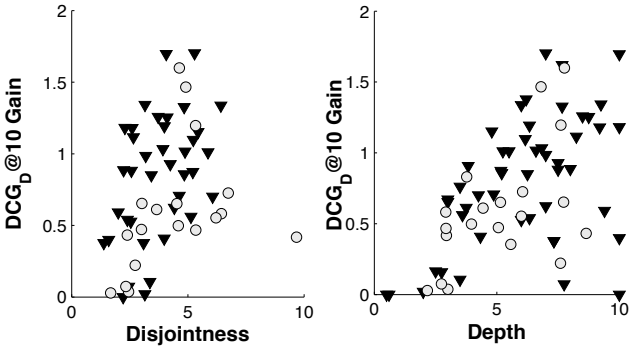

Figure 5: Adaptivity Gain of DynamicMyopic as a function of Disjointness and Depth.

Using Prec@k as an example, the dynamic rankings improve over the (best possible) static ranking by about 15 to 20 percentage points. To our surprise, there is no significant difference in the quality of the dynamic rankings produced by DynamicMyopic and DynamicLookahead.

6.2 Experiment 2: Influence Factors

To better understand what types of queries benefit most from dynamic ranking, consider the plots in Figure 5, both of which show the adaptivity gain of DynamicMyopic on the y-axis. Each circle represents the adaptivity gain of a particular Interactive query, while each triangle represents the adaptivity gain of a particular Web query. The x-axis of Figure 5 (left) shows the "disjointness" of the relevance profiles of a query, which is defined as follows. Denote with $Rel(q)$ the set of documents that are relevant to some profile r of q. Then disjointness is defined as the negative logarithm of the probability that a randomly selected document from $Rel(q)$ will be relevant to two relevance profiles sampled according to $P(R = r|q)$. Clearly, if there is only a single profile (or multiple profiles that are very similar), disjointness will be (close to) 0. Since this is a non-ambiguous query, the adaptivity gain is low or 0. As can be seen in Figure 5, the adaptivity gain generally increases with increasing disjointness for both datasets. Large adaptivity gains can therefore be expected especially for highly ambiguous queries.

However, there is a second factor that influences the adaptivity gain. If all relevance profiles are disjoint and have only a single relevant document, there is never a reason to expand any documents, and it is easy to see that the adaptivity gain is zero. Figure 5 (right) plots adaptivity gain of $DCG@k$ against "depth", which we define as $\sum_r \min\{k, |r|\}P(r|q)$. The results show that dynamic ranking is most beneficial when the depth of the profiles is sufficiently large.

6.3 Experiment 3: Noisy User Behavior

The previous experiments assumed that users behave deterministically according to the policy π_{det}, meaning that they always expand relevant results and they never expand a non-relevant result. In practice, user behavior will likely be noisier, and we now explore how different levels of noise influence the adaptivity gain.

To model noisy user behavior, consider the family of user policies π_ϵ where users expand a relevant link with probability $1 - \epsilon$ and expand a non-relevant link with probability ϵ. Figures 6 and 7 show the adaptivity gain for DCG@10 for

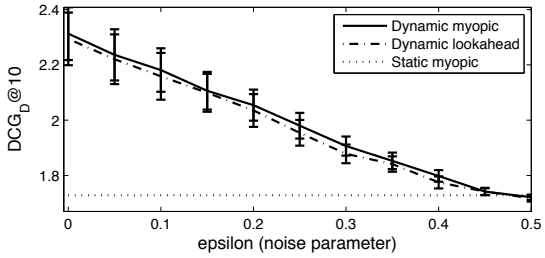

Figure 6: Adaptivity Gain depending on noise in user behavior on the Interactive dataset.

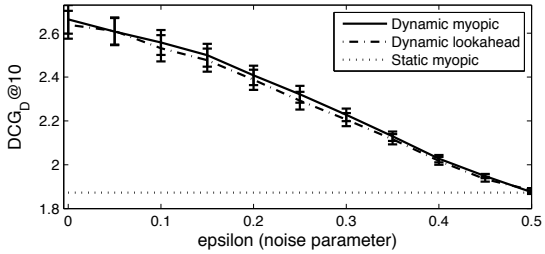

Figure 7: Adaptivity Gain depending on noise in user behavior on the Web dataset.

different values of ϵ used both during the generation of the ranking tree, and for evaluation. Adaptivity gain degrades gracefully and remains substantial even for high levels of noise. As expected, the adaptivity gain must be zero for $\epsilon = 0.5$, since the user interactions are random and do not provide any information about the user's intent.

Over all noise levels, there is no significant difference between DynamicMyopic and DynamicLookahead. We therefore conclude that DynamicMyopic is preferable in practical applications due to its superior efficiency.

7. LEARNING CONDITIONAL RELEVANCES

The previous experiments assumed that the distribution of relevance profiles is known. But when the retrieval algorithms are applied to new queries, one must estimate the unconditional expected relevances $P(r_d|q)$ to apply StaticMyopic, as well as the conditional expected relevances $P(r_d|q, C_\sigma = \gamma, \pi)$ to apply DynamicMyopic and DynamicLookahead for modular performance measures.

We now provide proof-of-concept that these quantities can be learned from data. We split the learning problem into three different logistic regression models: one which predicts $P(r_d|q, C_\sigma = \emptyset)$, one for $P(r_d|q, \pi, C_\sigma = \gamma)$ where γ consists entirely of "skips", and one for $P(r_d|q, \pi, C_\sigma = \gamma)$ with mixed "expands" and "skips". We train those models using the LR-TRIRLS software.[4]

Feature Generation. For both the Interactive and the Web datasets, we first represent each document as a set of fields in the vector space model. For Interactive, these fields consist of the headline, full text, byline combined with dateline, and a "summary abstract" consisting of the first 100 words of the full text combined with any remaining metadata.

For Web, the fields are the title, full text, URL combined with author, and the "summary abstract."

[4] `http://komarix.org/ac/lr`

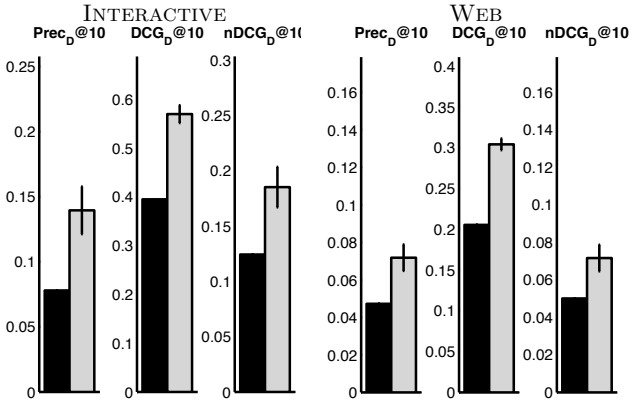

Figure 8: Leave-one-out performance of StaticMyopic (black) and DynamicMyopic (gray) on the Interactive (left) and Web (right) datasets. The error-bars indicate standard error of the adaptivity gain.

In the $P(r_d|q)$ model, each document is represented by a set of features which relate it to the query (e.g. the cosine similarity between each document field and the query). While the second two logistic regression models also include these features, they also utilize features which relate the document to

the current context C_σ (e.g. the TFIDF cosine similarities between the fields of the candidate document and the set of expanded documents).

Training and Evaluation. We evaluate using leave-one-query-out validation and left all parameters of the LR-TRIRLS software at their default settings. Given the previous results, we focus on evaluating the adaptivity gain of DynamicMyopic optimized for DCG@10, with π_{det} as the user interaction policy. The candidate sets $\mathcal{D}$ consist of all judged documents for Interactive, and all documents judged relevant to some profile for Web. For Web, we further restrict the candidate sets to the 50 million document "WebB" subset for efficiency reasons.

To generate training data, since the number of candidate documents that are irrelevant to a particular profile far outweigh the number of relevant documents, we randomly subsample training examples to get a ratio of 1 relevant to 9 irrelevant examples. For the conditional models, we uniformly choose among examples with 1,2,3,4,5 "expands", and 0,1,2,3,4,5,10 "skips", generating $3,333$ training examples for each of the three logistic regression models.

Results. The results of StaticMyopic and DynamicMyopic are given in Figure 8. On both datasets, the average performance of the dynamic model is higher than the static model for all combinations of datasets and evaluation measures. Despite the small sample size of only 20 and 50 queries respectively, the difference is significant at the 95%-level using a paired two-tailed Wilcoxon significance test in all cases.

These results show that a simple linear logistic regression approach can sufficiently model the conditional expected relevances to get a substantial adaptivity gain. We conjecture that more sophisticated machine learning methods specially designed for estimating $P(r_d|q, \pi, C_\sigma = \gamma)$ will further improve the gain. It would also be interesting to learn $P(r_d|q, \pi, C_\sigma = \gamma)$ directly from usage data.

8. DISCUSSION AND FUTURE WORK

We believe that the results presented in this paper make a convincing case that dynamic ranked retrieval is a promising area for future work. Dynamic ranked retrieval provides an approach for overcoming the inherently limited performance of using single static rankings for ambiguous queries. For instance, for web search, we conjecture that it is much easier to improve search quality by another 5 percentage points in nDCG by moving to a dynamic retrieval model, instead of trying to further optimize a (probably already close to optimal) static ranking function.

Dynamic ranked retrieval occupies a "sweet spot" in the spectrum of interactive retrieval where theoretical simplicity, computational tractablility, and impact on user experience meet. With respect to user experience, dynamic ranked retrieval does not require (but can accommodate) radically different user interfaces. In fact, the first result the user receives is a traditional-looking ranking, and our dynamic ranked retrieval algorithms merely provide a well-founded method for diversification. This makes the dynamic ranked retrieval system usable even if users decide to never expand any results. Nevertheless, we believe that users will easily understand the inherent semantics of dynamic ranked retrieval, since they are already familiar with navigating drop-down menus and result indentation in other contexts.

Many additional questions remain to be answered. From a theoretical standpoint, finding the optimal policy tree is computationally hard [13], but a more interesting direction may be to investigate which retrieval measures and dynamic ranking algorithms can provide stronger performance guarantees over the optimal static ranking. The range of possible user interaction policies π also requires further exploration, both in terms of additional policies as well as their verification against actual user behavior. For example, our framework assumes that user interests remain static throughout the search session, which may often not hold in practice.

The broader goal is to design general retrieval frameworks that can model increasingly richer interactive settings for a wide variety of retrieval applications. Progress towards this goal requires understanding and incorporating other forms of implicit feedback [14]. It also requires modeling the relative benefits of different types of recommendations [15], which leads to the more general problem of jointly modeling user interface design and content relevance.

9. CONCLUSIONS

This paper proposed a dynamic ranked retrieval model which allows users to interactively expand the ranking to further refine the information need. The model is based on a concise decision-theoretic framework that naturally generalizes both the standard and the intent-aware static retrieval models. The framework provides a principled way of evaluating dynamic retrieval systems, as well as a basis for deriving dynamic ranked retrieval algorithms. We presented two such algorithms and prove theoretical guarantees for their retrieval quality. We also evaluated the algorithms empirically and find that dynamic rankings can provide very substantial gains in retrieval performance. Finally, we showed that the retrieval functions of these algorithms can be learned from training data.

10. ACKNOWLEDGMENTS

This work was funded in part by NSF Award IIS-0905467. The third author was also funded in part by a Microsoft Research Graduate Fellowship. The authors thank Robert Kleinberg for valuable discussions regarding this work.

11. REFERENCES

[1] I. J. Aalbersberg. Incremental relevance feedback. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 11–22, 1992.

[2] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *ACM Conference on Web Search and Data Mining (WSDM)*, 2009.

[3] A. Anagnostopoulos, L. Becchetti, C. Castillo, and A. Gionis. An optimization framework for query recommendation. In *ACM Conference on Web Search and Data Mining (WSDM)*, 2010.

[4] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart: Trec 3. In *Text REtrieval Conference (TREC)*, 1994.

[5] G. Cao, J.-Y. Nie, J. Gao, , and S. Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2008.

[6] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2008.

[7] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and reproducing summaries. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 1998.

[8] C. Clarke, M. Kolla, G. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. Mackinnon. Novelty and diversity in information retrieval evaluation. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2008.

[9] M. Cramer, M. Wertheim, and D. Hardtke. Demonstration of improved search result relevancy using real-time implicit relevance feedback. In *SIGIR Workshop on Understanding the User – Logging and Interpreting User Interactions in Information Search and Retrieval*, 2009.

[10] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. A framework for selective query expansion. In *ACM Conference on Information and Knowledge Management (CIKM)*, 2004.

[11] K. El-Arini, G. Veda, D. Shahaf, and C. Guestrin. Turning down the noise in the blogosphere. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2009.

[12] N. Fuhr. A probability ranking principle for interactive information retrieval. *Information Retrieval*, 11(3):251 – 265, 2008.

[13] D. Golovin and A. Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. In *Conference on Learning Theory (COLT)*, 2010.

[14] Q. Guo and E. Agichtein. Ready to buy or just browsing? Detecting web searcher goals from interaction data. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2010.

[15] D. Kelly, K. Gyllstrom, and E. W. Bailey. A comparison of query and term suggestion features for interactive searching. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2009.

[16] J. Koren, Y. Zhang, and X. Liu. Personalized interactive faceted search. In *World Wide Web Conference (WWW)*, pages 477–486, 2008.

[17] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[18] J. Ponte. Language models for relevance feedback. In *Advances in Information Retrieval*, pages 73–96. Springer, 2000.

[19] F. Radlinski, P. Bennett, B. Carterette, and T. Joachims. Redundancy, diversity, and interdependent document relevance, a summary of the sigir 2009 workshop. *ACM SIGIR Forum*, 2009.

[20] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *International Conference on Machine Learning (ICML)*, 2008.

[21] S. Robertson. The probability ranking principle in ir. journal of documentation. *Journal of Documentation*, 33(4):294–304, 1977.

[22] J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.

[23] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review*, 18(2):95–145, 2003.

[24] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society of Information Science*, 41(4):288–297, 1990.

[25] X. Shen and C. Zhai. Active feedback in ad hoc information retrieval. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2005.

[26] Z. Xu and R. Akella. Active relevance feedback for difficult queries. In *ACM Conference on Information and Knowledge Management (CIKM)*, 2008.

[27] Y. Yue and T. Joachims. Predicting diverse subsets using structural svms. In *International Conference on Machine Learning (ICML)*, 2008.

[28] C. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: Methods and evaluation metric for subtopic retrieval. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2003.

[29] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *ACM Conference on Information and Knowledge Management (CIKM)*, 2001.

[30] L. Zhang and Y. Zhang. Interactive retrieval based on faceted feedback. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2010.