Structured Learning of Two-Level Dynamic Rankings

Karthik Raman
Dept. of Computer Science
Cornell University
Ithaca NY 14850 USA
karthik@cs.cornell.edu

Thorsten Joachims
Dept. of Computer Science
Cornell University
Ithaca NY 14850 USA
tj@cs.cornell.edu

Pannaga Shivaswamy
Dept. of Computer Science
Cornell University
Ithaca NY 14850 USA
pannaga@cs.cornell.edu

ABSTRACT

For ambiguous queries, conventional retrieval systems are bound by two conflicting goals. On the one hand, they should diversify and strive to present results for as many query intents as possible. On the other hand, they should provide depth for each intent by displaying more than a single result. Since both diversity and depth cannot be achieved simultaneously in the conventional *static* retrieval model, we propose a new *dynamic* ranking approach. In particular, our proposed *two-level* dynamic ranking model allows users to adapt the ranking through interaction, thus overcoming the constraints of presenting a one-size-fits-all static ranking. In this model, a user's interactions with the first-level ranking are used to infer this user's intent, so that second-level rankings can be inserted to provide more results relevant to this intent. Unlike previous dynamic ranking models, we provide an algorithm to efficiently compute dynamic rankings with provable approximation guarantees. We also propose the first principled algorithm for learning dynamic ranking functions from training data. In addition to the theoretical results, we provide empirical evidence demonstrating the gains in retrieval quality over conventional approaches.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Theory

Keywords

Diversified Retrieval, Structured Learning, Submodular Optimization, Web Search, Ranked Retrieval

1. INTRODUCTION

Search engine users often express different information needs using the same query. For such *ambiguous queries*, a single query can represent multiple intents – ranging from coarse (*e.g.*, queries such as *apple*, *jaguar* and *SVM*) to fine-grained ambiguity (*e.g.,* the query *apple ipod* with the intent of either buying the device or reading reviews).

Conventional retrieval methods do not explicitly model query ambiguity, but simply rank documents by their probability of relevance independently for each result [9]. A major limitation of this approach is that it favors results for the most prevalent intent. In the extreme, the retrieval system focuses entirely on the prevalent intent, but produces no relevant results for the less popular intents. Diversification-based methods (e.g. [3, 13, 4, 12, 10]) try to alleviate this problem by including at least one relevant result for as many intents as possible. However, this necessarily leads to fewer relevant results for each intent. Clearly, there is an inherent trade-off between *depth* (number of results provided for an intent) and *diversity* (number of intents served) in the conventional ranked-retrieval setting, since increasing one invariably leads to a decrease of the other. How can we avoid this trade-off and obtain diversity without compromising depth?

We argue that a key to solving the conflict between depth and diversity lies in the move to *dynamic* retrieval models [2] that can take advantage of user interactions. Instead of presenting a single one-size-fits-all ranking, dynamic retrieval models allow users to adapt the ranking dynamically through interaction, as is done by surfcanyon.com [5]. Brandt et al. [2] have already given theoretical and empirical evidence that even limited interactivity can greatly improve retrieval effectiveness. However, Brandt et al. [2] did not provide an efficient algorithm for computing dynamic rankings with provable approximation guarantees, nor did they provide a principled algorithm for learning dynamic ranking functions. In this paper, we resolve these open questions.

In particular, we propose a new *two-level dynamic ranking model*. Intuitively, the first level provides a diversified ranking of results on which the system can sense the user's interactions. Conditioned on this feedback, the system then interactively provides a second-level rankings. A possible layout is given in Figure 1. The left-hand panel shows the first-level ranking initially presented to the user. The user then chooses to expand the second document (*e.g.* by clicking) and a second-level ranking is inserted as shown in the right panel. Conceptually, the retrieval system maintains two levels of rankings, where each second-level ranking is conditioned on the head document in the first-level ranking.

To operationalize the construction and learning of two-level rankings in a rigorous way, we define a new family of submodular performance measure for diversified retrieval. Many existing retrieval measures (e.g., Precision@k, DCG, Intent Coverage) are special cases of this family. We then

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

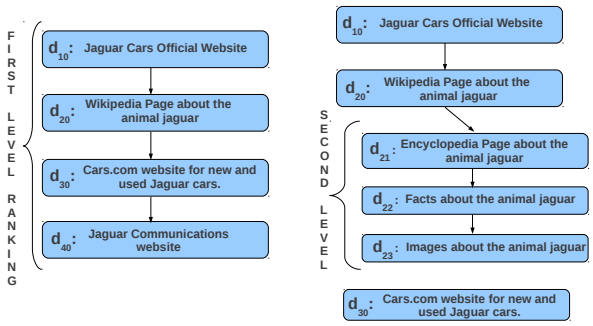

Figure 1: A user interested in the animal "jaguar" interacts with the first-level ranking (left) and obtains second-level results (right).

operationalize the problem of computing an optimal two-level ranking as maximizing the given performance measure. While this optimization problem is NP-hard, we provide an algorithm that has an $1-e^{-(1-\frac{1}{e})}$ approximation guarantee.

Finally, we also propose a new method for learning the (mutually dependent) relevance scores needed for two-level rankings. Following a structural SVM approach, we learn a discriminant model that resembles the desired performance measure in structure, but learns to approximate unknown intents based on query and document features. This method generalizes the learning method from [12] to two-level rankings and a large class of loss functions.

2. TWO-LEVEL DYNAMIC RANKINGS

Current methods for diversified retrieval are *static* in nature, *i.e.,* they stay unchanged through a user session. On the other hand, a *dynamic* model can adapt the ranking based on interactions with the user. The primary motivation for using a dynamic model is addressing the inherent trade-off between depth and diversity in static models.

Consider the example with four (equally likely) user intents $\{t_1, ..., t_4\}$ and documents $\{d_1, ..., d_9\}$ with relevance judgments $U(d_j|t_i)$ as given in Table 1. On the one hand, a non-diversified static ranking method could present $d_7 \rightarrow d_8 \rightarrow d_9$ as its top three documents, providing two relevant documents for intents t_3 and t_4 but none for intents t_1 and t_2. On the other hand, a diversified static ranking $d_7 \rightarrow d_1 \rightarrow d_4$ covers all intents, but this ranking lacks depth since no user gets more than one relevant document.

As an alternative, consider the following two-level dynamic ranking. The user is presented with $d_7 \rightarrow d_1 \rightarrow d_4$ as the first-level ranking. Users can now expand any of the first-level results to view a second-level ranking. Users interested in d_7 (and thus having intent t_3 or t_4) can *expand* that result and receive a second-level ranking consisting of d_8 and d_9. Similarly, users interested in d_1 will get d_2 and d_3; and users interested in d_4 will get d_5 and d_6.

For this dynamic ranking, every user gets at least one relevant result after scanning at most three documents (i.e. the first-level ranking). Furthermore, users with intents t_3 and t_4 receive two relevant results in the top three positions of their dynamically constructed ranking $d_7 \rightarrow d_8 \rightarrow d_9 \rightarrow d_1 \rightarrow d_4$. Similarly, users with intent t_1 also receive two relevant results in the top three positions while those with intent t_2 still receive one relevant result. This illustrates how a dynamic two-level ranking can simultaneously provide diversity and increased depth.

$U(d_j\|t_i)$	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
t_1	1	1	1	0	0	0	0	0	0
t_2	0	0	0	1	1	1	0	0	0
t_3	0	0	0	0	0	0	1	1	0
t_4	0	0	0	0	0	0	1	0	1

Table 1: Utility $U(d_j|t_i)$ of document d_j for intent t_i.

In the above example, interactive feedback from the user was the key to achieving both depth and diversity. More generally, we assume the following **model of user behavior**, which we denote as policy π_d. Users expand a first-level document if and only if that document is relevant to their intent. When users skip a document, they continue with the next first-level result. When users expand a first-level result, they go through the second-level rankings before continuing from where they left off in the first-level ranking. It is thus possible for a user to see multiple second-level rankings. Hence we do not allow documents to appear more than once across all two-level rankings.

Note that this user model differs from the one proposed in [2] in several ways. First, it assumes feedback only for the first-level ranking, while the model in [2] requires that users give feedback many levels deep. Second, unlike in [2], we model that users return to the top-level ranking. We conjecture that these differences make the two-level model more natural and appropriate for practical use.

We now define some notation used in this paper. The documents shown in a first-level ranking of **length** L are called the **head** documents. The documents shown in a second-level ranking are called the **tail** documents. The number of tail documents is referred to as the **width** W. A **row** denotes a **head** document and all its **tail** documents. Static rankings are denoted as θ while two-level rankings are denoted as $\Theta = (\Theta_1, \Theta_2, ...\Theta_i, ..)$. Here $\Theta_i = (d_{i0}, d_{i1}, ..., d_{ij}, ...)$ refers to the i^{th} row of a two-level ranking, with d_{i0} representing the head document of the row and d_{ij} denoting the j^{th} tail document of the second-level ranking. We denote the candidate set of documents to rank for a query q by $\mathcal{D}(q)$, the set of possible intents by $\mathcal{T}(q)$ and the distribution over an intent $t \in \mathcal{T}(q)$, given a query q, by $\mathbf{P}[t|q]$. Unless otherwise mentioned, dynamic ranking refers to a two-level dynamic ranking in the following.

3. PERFORMANCE MEASURES FOR DIVERSIFIED RETRIEVAL

To define what constitutes a good two-level dynamic ranking, we first define the measure of retrieval performance we would like to optimize. We then design our retrieval algorithms and learning methods to maximize this measure. In the following, we start with evaluation measures for one-level rankings, and then generalize them to the two-level case.

3.1 Measures for Static Rankings

Existing performance measures range from those that do not explicitly consider multiple intents (e.g. NDCG, Avg Prec), to measures that reward diversity. Measures that reward diversity give lower marginal utility to a document, if the intents the document is relevant to are already well represented in the ranking. We call this the *diminishing returns* property. The extreme case is the "intent coverage" measure (e.g. [10, 12]), which attributes utility only to the first document relevant for an intent.

We now define a family of measures that includes a whole

range of diminishing returns models, and that includes most existing retrieval measures. Let $g : \mathbb{R} \to \mathbb{R}$ with $g(0) = 0$ be a concave, non-negative, and non-decreasing function that models the diminishing returns, then we define the utility of the ranking $\theta = (d_1, d_2, ..., d_k)$ for intent t as

$$U_g(\theta|t) = g\Big(\sum_{i=1}^{|\theta|} \gamma_i U(d_i|t)\Big). \quad (1)$$

The $\gamma_1 \geq ... \geq \gamma_k \geq 0$ are discount factors and $U(d|t)$ is the relevance rating of document d for intent t. For a distribution of user intents $\mathbf{P}[t|q]$ for query q, the overall utility of a static ranking θ is the expectation

$$U_g(\theta|q) = \sum_{t \in \mathcal{T}(q)} \mathbf{P}[t|q]\, U_g(\theta|t). \quad (2)$$

Note that many existing retrieval measures are special cases of this definition. For example, if one chose g to be the identity function, one recovers the intent-aware measures proposed in [1] and the modular measures defined in [2]. Further restricting $\mathbf{P}[t|q]$ to put all probability mass on a single intent leads to conventional measures like DCG for appropriately chosen γ_i. At the other extreme, chosing $g(x) = \min(x, 1)$ leads to the intent coverage measure [10, 12]. Since g can be chosen from a large class of functions, this family of performance measures covers a wide range of diminishing returns models.

3.2 Measures for Dynamic Rankings

We now extend our family of performance measures to dynamic rankings. The key change for dynamic rankings is that users interactively adapt which results they view.

How users expand first-level results was defined in Section 2 as π_d. Under π_d, it is natural to define the utility of a dynamic ranking Θ analogous to Equation (1).

$$U_g(\Theta|t) = g\Big(\sum_{i=1}^{|\Theta|}\Big(\gamma_i U(d_{i0}|t) + \sum_{j=1}^{|\Theta_i|} \gamma_{ij} U(d_{i0}|t) U(d_{ij}|t)\Big)\Big). \quad (3)$$

Like for static rankings, $\gamma_1 \geq \gamma_2 \geq ...$ and $\gamma_{i1} \geq \gamma_{i2} \geq ...$ are position-dependent discount factors. Furthermore, we again take the expectation over multiple user intents as in Equation (2) to obtain $U_g(\Theta|q)$.

Note that the utility of a second-level ranking for a given intent is zero unless the head document in the first-level ranking has non-zero relevance for that intent. This encourages second-level rankings to only contain documents relevant to the same intents as the head document, thus providing depth. The first-level ranking, on the other hand, provides diversity as controlled through the choice of function g. The "steeper" g diminishes returns of additional relevant documents, the more diverse the first-level ranking gets.

4. COMPUTING DYNAMIC RANKINGS

In this section, we provide an efficient algorithm for computing dynamic rankings that maximize the performance measures defined in the previous section. In the proposed greedy Algorithm 1, the operator $\oplus$ denotes either adding a document to a row, or adding a row to an existing ranking. In each iteration, Algorithm 1 considers every document in the remaining collection as the head document of a candidate row. For each candidate row, W documents are greedily added to maximize the utility $U_g(\Theta|q)$ of the resulting partial dynamic ranking Θ. Once rows of length W are constructed, the row which maximizes the utility is added to the ranking. The above steps are repeated until the ranking has L rows. Algorithm 1 is efficient, requiring $O(|\mathcal{T}|)$ space and $O(|\mathcal{T}||\mathcal{D}|^2)$ time.

Algorithm 1 for computing a two-level dynamic ranking.

Input: $(q, \mathcal{D}(q), \mathcal{T}(q), \mathbf{P}[t|q] : t \in \mathcal{T}(q))$, $g(\cdot)$, L, W.
Output: A dynamic ranking Θ.
$\Theta \leftarrow$ new_two_level()
while $|\Theta| \leq L$ **do**
  $bestU \leftarrow -\infty$
  for all $d \in \mathcal{D}(q)$ s.t. $d \notin \Theta$ **do**
    $row \leftarrow$ new_row(); $row.head \leftarrow d$
    for $j = 1$ to W **do**
      $bestDoc \leftarrow \text{argmax}_{d' \notin \Theta \cup row} U_g(\Theta \oplus (row \oplus d')|q)$
      $row \leftarrow row \oplus bestDoc$
    if $U_g(\Theta \oplus row|q) > bestU$ **then**
      $bestU \leftarrow U_g(\Theta \oplus row|q)$; $bestRow \leftarrow row$
  $\Theta \leftarrow \Theta \oplus bestRow$

Our greedy algorithm is closely related to submodular function maximization. Maximizing monotonic submodular functions is a hard problem, but a greedily constructed set gives an $(1-1/e)$ approximation [7] to the optimal. Since the definition of our utility in (2) involves a concave function, it is not hard to show that selecting a ranking of rows is a submodular maximization problem. Moreover, given the head document, finding the best row is also a submodular maximization problem. Thus, finding a dynamic ranking to maximize our utility is a *nested* submodular maximization problem, and we can show the following approximation guarantee for Algorithm 1.

LEMMA 1. *Algorithm 1 is* $(1 - e^{-(1-\frac{1}{e})})$ *approximate.*

The proof can be found in [8]. It follows [6], but generalizes the result from max coverage to our more general setting.

5. LEARNING DYNAMIC RANKINGS

In the previous section, we showed that a dynamic ranking can be efficiently computed when all the intents and relevance judgments for a given query are known. In this section, we propose a supervised learning algorithm that can predict dynamic rankings on previously unseen queries.

Our goal here is to learn a mapping from a query q to a dynamic ranking Θ. We pose this as the problem of learning a weight vector $\mathbf{w} \in \mathbb{R}^N$ from which we can make a prediction as follows:

$$h_{\mathbf{w}}(q) = \underset{\Theta}{\text{argmax}}\ \mathbf{w}^\top \Psi(q, \Theta). \quad (4)$$

As further explained below, $\Psi(q, \Theta) \in \mathbb{R}^N$ is a joint feature-map between query q and dynamic ranking Θ.

Given a set of training examples $(q^i, \Theta^i)_{i=1}^n$, the structural SVM framework [11] can be use to learn a discriminant function by minimizing the empirical risk $\frac{1}{n}\sum_{i=1}^n \Delta(\Theta^i, h_{\mathbf{w}}(q^i))$, where Δ is a loss function. Unfortunately, however, the Θ^i are typically not given directly as part of the training data. Instead, we assuming that we are given training data of the form $(q^i, \mathcal{D}(q^i), \mathcal{T}(q^i), \mathbf{P}[t|q] : t \in \mathcal{T}(q^i))_{i=1}^n$, and we then compute the dynamic rankings Θ^i by maximizing the utility U_g (approximately) using Algorithm 1. These Θ^i's will be used as the training labels henceforth.

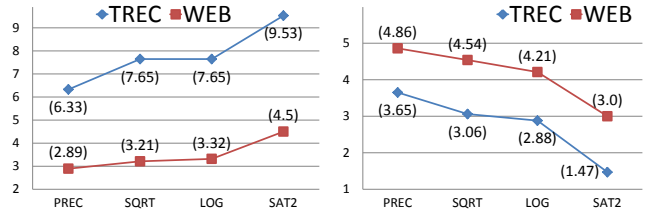

Figure 2: Average number of intents covered (left) and average number of documents for prevalent intent (right) in the first-level ranking.

A key aspect of structural SVMs is to appropriately define the joint-feature map $\Psi(q, \Theta)$. For our problem, we propose

$$\mathbf{w}^\top \Psi(q, \Theta) := \sum_{v \in V_{\mathcal{D}(q)}} \mathbf{w}_v^\top \phi_v U_g(\Theta|v) + \sum_{s \in V_{\mathcal{D}(q) \times \mathcal{D}(q)}} \mathbf{w}_s^\top \phi_s(\Theta), \quad (5)$$

where $V_{\mathcal{D}(q)}$ denotes an index set over the words in the candidate set $\mathcal{D}(q)$. The vector ϕ_v denotes word-level features (for example, how often a word occurs in a document) for the word corresponding to index v. The utility $U_g(\Theta|v)$ is analogous to (3) but is now over the words in the vocabulary (rather than over intents). The word-level features are reminiscent of the features used in diverse subset prediction [12]. The key assumption is that the words in a document are correlated with the intent since documents relevant to the same intent are likely to share more words than documents that are relevant to different intents.

The second term in Eq. 5 captures the similarity between head and tail documents. In this case, $V_{\mathcal{D}(q) \times \mathcal{D}(q)}$ denotes an index set over all document pairs in $\mathcal{D}(q)$. Consider an index s that corresponds to documents d_1 and d_2 in $\mathcal{D}(q)$. $\phi_s(\Theta)$ is a feature vector describing the similarity between d_1 and d_2 in Θ when d_1 is a head document in Θ and d_2 occurs in the same row as d_1 ($\phi_s(\Theta)$ is simply a vector of zeros otherwise). An example of a feature in $\phi_s(\Theta)$ that captures the similarity between two documents is their TFIDF cosine.

Using these features, $\mathbf{w}^\top \Psi(q, \Theta)$ models the utility of a given dynamic ranking Θ. During learning, $\mathbf{w}$ should be selected so that better rankings receive higher utility than worse rankings. This is achieved by solving the following structural SVM optimization problem for $\mathbf{w}$:

$$\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i \quad (6)$$

$$\text{s.t.} \forall i, \forall \Theta : \mathbf{w}^\top \Psi(q^i, \Theta^i) - \mathbf{w}^\top \Psi(q^i, \Theta) \geq \Delta(\Theta^i, \Theta|q^i) - \xi_i$$

The constraints in the above formulation ensure that the predicted utility for the target ranking Θ^i is higher than the predicted utility for any other Θ. The objective function in (6) minimizes the empirical risk while trading it off (via the parameter $C > 0$) with the margin. The loss between Θ^i and Θ is given by $\Delta(\Theta^i, \Theta|q^i) := 1 - \frac{U_g(\Theta|q^i)}{U_g(\Theta^i|q^i)}$ which ensures that the loss is zero when $\Theta = \Theta^i$. It is easy to see that a dynamic ranking Θ has a large loss when its utility is low compared to the utility of Θ^i.

Even though the Eq. (6) has an exponential number of constraints, the quadratic program in Eq. 6 can be solved in polynomial time using a cutting-plane algorithm [11]. In each iteration of the cutting-plane algorithm, the most violated constraints in (6) are added to a working set and the resulting quadratic program is solved. Given a current $\mathbf{w}$, the most violated constraints are obtained by solving:

$$\underset{\Theta}{\text{argmax}} \ \mathbf{w}^\top \Psi(q^i, \Theta) + \Delta(\Theta^i, \Theta|q^i). \quad (7)$$

Algorithm 1 can be used to solve the above problem, even though the formal approximation guarantee does not hold in this case. Once a weight vector $\mathbf{w}$ is obtained, the dynamic ranking for a test query can be obtained from Eq. (4).

6. EXPERIMENTS

Experiments were conducted on the TREC 6-8 Interactive Track (TREC) and the Diversity Track of TREC 18 using the ClueWeb collection (WEB). The 17 queries in TREC contain between 7 to 56 different manually judged intents. In the case of WEB, we used 28 queries with 4 or more intents. We consider the probability $\mathbf{P}[t]$ of an intent proportional to the number of documents relevant to that intent. A key difference between the two datasets is that the most prevalent intent covers 73.4% of all relevant documents for the WEB dataset, but only 37.6% for TREC.

The number of documents in the first-level ranking was set to 5. The width of the second-level rankings was set to 2. For simplicity, we chose all factors γ_i and γ_{ij} in Equations (1) and (3) to be 1. Further, we chose $U(d|t) = 1$ if document d was relevant to intent t and set $U(d|t) = 0$ otherwise.

More details about the experiments and additional results can also be found in [8].

6.1 Controlling Diversity and Depth

The key design choice of our family of utility measures is the concave function g. As Algorithm 1 directly optimizes utility, we explore how the choice of g affects various properties of the two-level rankings produced by our method.

We experiment with four different concave functions g, each providing a different diminishing-returns model. At one extreme, we have the identity function $g(x) = x$ which corresponds to modular returns. Using this function in Eq. (1) leads to the intent-aware Precision measure proposed in [1], and it is the only function considered in [2]. We therefore refer to this function as PREC. It is not hard to show that Algorithm 1 actually computes the optimal two-level ranking for this choice of g. On the other end of the spectrum, we study $g(x) = \min(x, 2)$. By remaining constant after two, this function discourages presenting more than two relevant documents for any intent. This measure will be referred to as SAT2 (short for "satisfied after two"). In between these two extremes, we study the square root function (SQRT) $g(x) = \sqrt{x}$ and the log function (LOG) $g(x) = \log(1 + x)$.

To explore how dynamic rankings can differ, we used Algorithm 1 to compute the two-level rankings (approximately) maximizing the respective measure. Figure 2 shows how g influences diversity. The left-hand plot shows how many different intents are represented in the top 5 results of the first-level ranking on average. The graph shows that the stronger the diminishing-returns model, the more different intents are covered in the first-level ranking. In particular, the number of intents almost doubles on both datasets when moving from PREC to SAT2. In contrast, the number of documents on the most prevalent intent in the first-level ranking decreases, as shown in the right-hand plot. This illustrates how the choice of g can be used to control the desired amount of diversity in the first-level ranking.

Table 2 provides further insight into the impact of g, now also including the contributions of the second-level rankings.

Eval. \ Optim.	PREC	SQRT	LOG	SAT2
PREC	**0.315**	0.302	0.294	0.164
SQRT	1.612	**1.664**	1.659	1.333
LOG	1.216	1.267	**1.27**	1.046
SAT2	1.18	1.335	1.349	**1.487**

Table 2: Performance when optimizing and evaluating using different performance measures for TREC.

The rows correspond to different choices for g when evaluating expected utility according to Eq. (3), while the columns show which g the two-level ranking was optimized for. Not surprisingly, the diagonal entries of Tables 2 show that the best performance for each measure is obtained when optimizing for it. The off-diagonal entries show that different g used during optimization lead to substantially different rankings. This is particularly apparent when optimizing the two extreme performance measures PREC and SAT2; optimizing one invariably leads to rankings that have a low value of the other. In contrast, optimizing LOG or SQRT results in much smoother behavior across all measures, and both seem to provide a good compromise between depths (for the prevalent intent) and diversity. The results for WEB are qualitatively similar and are omitted for space reasons.

6.2 Static vs. Dynamic Ranking

The ability to simultaneously provide depth and diversity was a key motivation for our dynamic ranking approach. We now evaluate whether this goal is indeed achieved. We compare the two-level rankings produced by Algorithm 1 (denoted *Dyn*) with several static baselines. These static baselines are also computed by Algorithm 1, but with the width of the second-level rankings to 0.

First, we compare against a diversity-only static ranking that maximizes intent coverage as proposed in [12] (denoted *Stat-Div*). Second, we compare against a depth-only static ranking by chosing g to be the identity function (denoted *Stat-Depth*). And, third, we produce static rankings that optimize SQRT, LOG, and SAT2 (denoted *Stat-Util*). Note that both Dyn and Stat-Util optimize the same measure that is used for evaluation.

To make a fair comparison between static and dynamic rankings, we measure performance in the following way. For static rankings, we compute performance using the expectation of Eq. (1) at a depth cutoff of 5. In particular, we measure PREC@5, SQRT@5, LOG@5 and SAT2@5. For two-level rankings, the number of results viewed by a user depends on how many results he or she expands. So, we truncate any user's path through the two-level ranking after visiting 5 results and compute PREC@5, SQRT@5, LOG@5 and SAT2@5 for the truncated path.

Results of these comparisons are shown in Figure 3. First, we see that both Dyn and Stat-Util outperform Stat-Div, illustrating that optimizing rankings for the desired evaluation measure leads to much better performance than using a proxy measure as in Stat-Div. Note that Stat-Div never tries to present more than one result for each intent, which explains the extremely low "depth" performance in terms of PREC@5. But Stat-Div is not competitive even for SAT2, since it never tries to provide a second result.

Second, at first glance it may be surprising that Dyn outperforms Stat-Depth even on PREC@5, despite the fact that Stat-Depth explicitly (and globally optimally) optimizes depth.

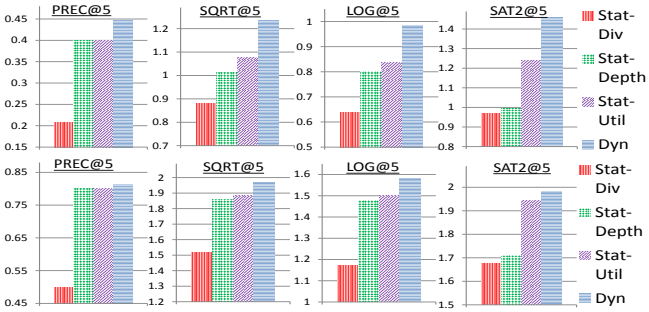

Figure 3: Comparing the retrieval quality of Static vs. Dynamic Rankings for TREC and WEB.

To understand consider the following situation where A is the prevalent intent, and there are three documents relevant to A and B and three relevant to A and C. Putting those sets of three documents into the first two rows of the dynamic ranking provides better PREC@5 than sequentially listing them in the optimal static ranking.

Overall we find the dynamic ranking method outperforming all static ranking schemes on all the metrics – in many cases with a substantial margin. This gain is more pronounced for TREC than for WEB. This can be explained by the fact that WEB queries are less ambiguous, since the single most prevalent intent accounts for more than 70% of all queries on average.

6.3 Learning Two-level Ranking Functions

So far we have evaluated how far Algorithm 1 can construct effective two-level rankings if the relevance ratings are known. We now explore how far our learning algorithm can predict two-level rankings for previously unseen queries. For all experiments in this section, we learn and predict using SQRT as the choice for g, since it provides a good trade-off between diversity and depth as shown above.

We performed standard preprocessing such as tokenization, stopword removal and Porter stemming. Since the focus of our work is on diversity and not on relevance, we rank only those documents that are relevant to at least one intent of a query. This simulates a candidate set that may have been provided by a conventional retrieval method. This setup is similar to that used by Yue and Joachims [12].

Many of our features in ϕ_v follow those used in [12]. These features provide information about the importance of a word in terms of two different aspects. A first type of feature describes the overall importance of a word. A second type of feature captures the importance of a word in a document. An example of this type of feature is whether a word appears with frequency at least y% in the document? Finally, we also use features ϕ_s that model the relationship between the documents in the second-level ranking and the corresponding head document of that row. Examples of this type of feature are binned features representing TFIDF similarity of document pairs and the number of common words that appear in both documents with a frequency at least x%.

Dynamic vs. Static: In the first set of experiments, we compare our learning method (*Dyn-SVM*) for two-level rankings with two static baselines. The first static baseline is the learning method from [12] which optimizes diversity (referred to as *Stat-Div*). It is one of the very few learning methods for learning diversified retrieval functions, and was shown to outperform non-learning methods like Essen-

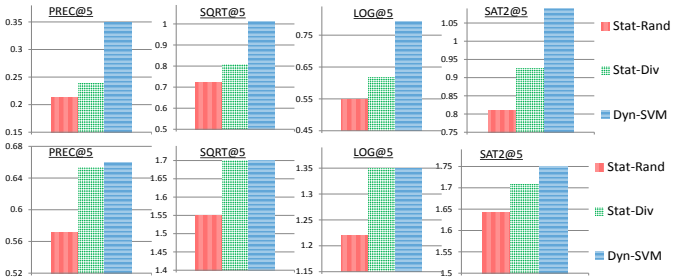

Figure 4: Performance of learned functions, comparing static & dynamic rankings for TREC and WEB.

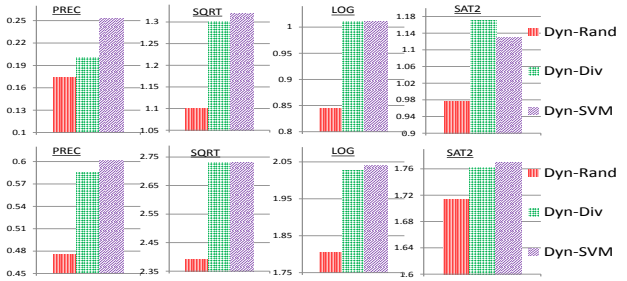

Figure 5: Comparing learned dynamic rankings with heuristic baselines for TREC and WEB.

tial Pages [10]. We also consider a *random* static baseline (referred to as *Stat-Rand*), which randomly orders the candidate documents. This is a competent baseline, since all our candidate documents are relevant to at least one intent.

Figure 4 shows the comparison between static and dynamic rankings. For TREC, *Dyn-SVM* substantially outperforms both static baselines across all performance metrics, mirroring the results we obtained in Section 6.2 where the relevance judgments were known. This shows that our learning method can effectively generalize the multi-intent relevance judgments to new queries. On the less ambiguous WEB dataset the differences between static and dynamic rankings are smaller. While *Dyn-SVM* substantially outperforms *Stat-Rand*, *Stat-Div* is quite competitive on WEB.

Learning vs. Heuristic Baselines: We also compare against alternative methods for constructing two-level rankings. We extend the static baselines *Stat-Rand* and *Stat-Div* using the following heuristic. For each result in the static ranking, we add a second-level ranking using the documents with the highest TFIDF similarity from the candidate set. This results in two dynamic baselines, which we call *Dyn-Rand* and *Dyn-Div*. The results are shown in Figure 5.

Since we compare two-level rankings of equal size, we measure performance in terms of expected utility. On both datasets *Dyn-SVM* performs substantially better than *Dyn-Rand*. This implies that our method can effectively learn which documents to place at the top of the first-level ranking. Surprisingly, simply extending the diversified ranking of *Dyn-Div* using the TFIDF heuristic produces dynamic rankings are are competitive with *Dyn-SVM*. In retrospect, this is not too surprising for two reasons. First, our experiments with *Dyn-SVM* use rather simple features to describe the relationship between the head document and documents in the second-level ranking – most of which are derived from their TFIDF similarity. Stronger features exploiting document ontologies or browsing patterns could easily be incorporated into the feature vector. Second, the learning method of *Dyn-Div* is actually a special case of *Dyn-SVM* when using the SAT1 loss (i.e. satisfied after a single relevant document) and second-level rankings of width 0. However, we argue that it is still highly preferable to directly optimize the desired loss function and two-level ranking using *Dyn-SVM*, since the reliance on heuristics may fail on other datasets.

7. CONCLUSIONS

We proposed a two-level dynamic ranking approach that provides both diversity and depth for ambiguous queries by exploiting user interactivity. In particular, we showed that the approach has the following desirable properties. First, it covers a large family of performance measures, making it easy to select a diminishing returns model for the application setting at hand. Second, we presented an efficient algorithm for constructing two-level rankings that maximizes the given performance measure with provable approximation guarantees. Finally, we provided a structural SVM algorithm for learning two-level ranking functions, showing that it can effectively generalize to new queries.

This work was funded in part under NSF Awards IIS-0905467, IIS-0713483, and IIS-0812091.

8. REFERENCES

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM*, 2009.

[2] C. Brandt, T. Joachims, Y. Yue, and J. Bank. Dynamic ranked retrieval. In *WSDM*, 2011.

[3] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, 1998.

[4] H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *SIGIR*, 2006.

[5] M. Cramer, M. Wertheim, and D. Hardtke. Demonstration of improved search result relevancy using real-time implicit relevance feedback. In *SIGIR*, 2009.

[6] D. S. Hochbaum and A. Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)*, 45:615–627, 1998.

[7] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Math Programming*, 14:265–294, 1978.

[8] K. Raman, P. Shivaswamy, and T. Joachims. Structured learning of two-level dynamic rankings. Arxiv 0298967, August 2011.

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

[10] A. Swamintahan, C. Metthew, and D. Kirovski. Essential pages. In *Technical Report, MSR-TR-2008-15*, Microsoft Research, 2008.

[11] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large-margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[12] Y. Yue and T. Joachims. Predicting diverse subsets using structural SVMs. In *ICML*, 2008.

[13] C. X. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *SIGIR*, 2003.