User Effort Minimization Through Adaptive Diversification

Mahbub Hasan Abhijith Kashyap Vagelis Hristidis Vassilis Tsotras
Computer Science and Engineering, UC Riverside
{hasanm, akash001, vagelis, tsotras}@cs.ucr.edu

ABSTRACT
Ambiguous queries, which are typical on search engines and recommendation systems, often return a large number of results from multiple interpretations. Given that many users often perform their searches on limited size screens (e.g. mobile phones), an important problem is which results to display first. Recent work has suggested displaying a set of results (Top-k) based on their relevance score with respect to the query and their diversity with respect to each other. However, previous works balance relevance and diversity mostly by a predefined fixed way. In this paper, we show that for different search tasks there is a different ideal balance of relevance and diversity. We propose a principled method for adaptive diversification of query results that minimizes the user effort to find the desired results, by dynamically balancing the relevance and diversity at each query step (e.g. when refining the query or viewing the next page of results). We introduce a navigation cost model as a means to estimate the effort required to navigate the query-results, and show that the problem of estimating the ideal amount of diversification at each step is NP-Hard. We propose an efficient approximate algorithm to select a near-optimal subset of the query results that minimizes the expected user effort. Finally we demonstrate the efficacy and efficiency of our solution in minimizing user effort, compared to state-of-the-art ranking methods, by means of an extensive experimental evaluation and a comprehensive user study on Amazon Mechanical Turk.

Categories and Subject Descriptors
H.2.m [**Database Management**]: Miscellaneous

Keywords
Diversity; User Effort Minimization; Mobile Applications

1. INTRODUCTION
Now-a-days mobile devices have become increasingly popular for searching. The majority of ecommerce sales are currently performed through smartphones or tablets, and specifically about 35% exclusively through smartphones [27]. Keyword-based interfaces have been widely adopted as the preferred query method because they are simple and intuitive. Nevertheless, keyword queries are often *ambiguous*, that is, they have multiple interpretations, and consequently generate many results. As an example, consider the query "memory" which might refer to computer memory (RAM, ROM, Flash etc.) or the song Memory from the acclaimed musical Cats. For such ambiguous queries, a ranking that considers only relevance (e.g., [8]) might return a large number of similar results from just one interpretation of the query, e.g. DDR3 RAM, and a user with different search intent (say music) might not find any result that is relevant to her in the first page (Top-k) of the results.

To deal with query ambiguity, recently several search applications have incorporated diversification while ranking results to improve the user experience. Examples of such applications include web search [4][6], recommendation systems [9][18][19][20], semi-structured databases [23], graph search [2], and document summarization [15]. A diversified ranking includes not only relevant (as judged by the underlying ranking function) results, but also results that may be less relevant and are diverse with respect to other results in the ranked list. By covering results from multiple interpretations of a query, diversified ranking thereby increases the probability of the user finding desired results based on her query intent [1]. Of course, just focusing on diversity and displaying the set of most diverse results is ineffective since some of these results may have low relevance. In its most general form, the problem of query result diversification is modeled as a bi-criteria optimization problem [1][5], which uses a trade-off parameter (λ) to tune the relative effect of relevance and diversity factors during ranking. Using λ, the impact of the diversity factor can be increased for highly ambiguous queries so as to include more diverse elements in the result set; whereas for very specific (non-ambiguous) queries, this factor can be decreased to prevent inclusion of results of lesser relevance.

As an example, consider Figure 1a which depicts the result set returned for the query 'Camera' (on a structured dataset like Amazon.com). As seen in the figure, the result set includes products from several categories including DSLR, Compact cameras and Accessories. Each result has a set of features (e.g. Brand, Megapixel, Zoom etc.). Note that the Lenses of DSLR cameras are considered in the Accessory category, therefore DSLR cameras do not have a Zoom feature. Since there is a limited paginated interface available to the user for displaying the results (mobile screen size etc.), Figures 1b-1d show the Top-3 results (first page) selected by varying the trade-off parameter between diversity and relevance. Note that the relevance ranking in this example assigns a higher score to DSLRs. For a user shopping for DSLR cameras, the ranking shown in Figure 1b, which prefers relevance over diversity, might be sufficient. However, a user looking for a camera Lens would prefer the highly diversified ranking shown in Figure 1d, where she could click on the Lens attribute value for attribute Type in the Accessory category to see more camera lenses.

KDD '14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623610

ID	Product	Category	Features		Rel
1	Camera	DSLR	Brand: Canon	Megapixel: 18.0	0.9
2	Camera	DSLR	Brand: Nikon	Megapixel: 16.0	0.8
3	Camera	DSLR	Brand: Canon	Megapixel: 14.0	0.7
4	Camera	DSLR	Brand: Nikon	Megapixel: 12.0	0.6
5	Camera	DSLR	Brand: Sony	Megapixel: 12.0	0.6
6	Camera	Compact	Brand: Panasonic	Zoom: 7x	0.6
			Megapixel: 14.0		
7	Camera	Compact	Brand: Panasonic	Zoom: 5x	0.6
			Megapixel: 16.0		
8	Camera	Compact	Brand: Fujifilm	Zoom: 5x	0.4
			Megapixel: 12.2		
9	Camera	Compact	Brand: Kodak	Zoom: 3x	0.2
			Megapixel: 10.0		
10	Camera	Accessory	Type: Lens	Focal Length: 18 – 55 mm	0.3
11	Camera	Accessory	Type: Lens	Focal Length: 55 – 300 mm	0.2

(a) Result Set

ID	Product	Category	Features		Rel
1	Camera	DSLR	Brand: Canon	Megapixel: 18.0	0.9
2	Camera	DSLR	Brand: Nikon	Megapixel: 16.0	0.8
3	Camera	DSLR	Brand: Canon	Megapixel: 14.0	0.7

(b) High Relevance and Low Diversity

ID	Product	Category	Features		Rel
1	Camera	DSLR	Brand: Canon	Megapixel: 18.0	0.9
2	Camera	DSLR	Brand: Nikon	Megapixel: 16.0	0.8
6	Camera	Compact	Brand: Panasonic	Zoom: 7x	0.6
			Megapixel: 14.0		

(c) Moderate Relevance and Moderate Diversity

ID	Product	Category	Features		Rel
1	Camera	DSLR	Brand: Canon	Megapixel: 18.0	0.9
6	Camera	Compact	Brand: Panasonic	Zoom: 7x	0.6
			Megapixel: 14.0		
10	Camera	Accessory	Type: Lens	Focal Length: 18 – 55 mm	0.3

(d) Low Relevance and High Diversity

Figure 1: A subset of results of the query *Camera* and associated ranked list of results

Note that, for a given query, the user navigation cost (the user "effort" or actions required to find the desired results) varies for different choices of the trade-off parameter (see Figure 2, for the query "Camera" using the *MMR* algorithm [10] to implement diversified ranking). Moreover, in Section 5 (Figure 7), we show experimentally, that the best value of the trade-off parameter λ varies for different queries. However, no previous work addresses the problem of computing a trade-off parameter that will minimize the user effort. Instead, many hard-code it to a reasonable value (fixing the relative weight between relevance and diversity). Recently, several methods have been proposed [4][16] to learn the trade-off parameter λ. Unfortunately, these methods depend on training data provided by the experts which are expensive to collect or might not be available. Further, they compute a single trade-off parameter for a query, whereas we show how this trade-off changes as the query refinement or results viewing progress.

Because of the display interface, finding the desired result to a particular query might involve several steps. If the user does not find her desired result on the first page, then she might take additional actions to find the result, such as: (a) scan additional pages looking for the results of interest, or, (b) refine the query by clicking on a displayed attribute value to focus on a subset of the original results. Our goal in this paper is to compute at each step a set of k results (corresponding to a page in the user's interface) that dynamically balances diversity and relevance such that the expected user navigation cost is minimized.

Figure 2: Navigation Cost vs. λ for Query *Camera*

What makes the problem difficult is that when the query is posed, neither the target result nor the sequence of actions the user will execute to find it, are known. Therefore, to compute the best set of results to display at each step, we must probabilistically consider all the unknown future user actions, which is a key challenge of our solution. For example, if the user poses the (highly ambiguous) query Camera while her target object is Lens, she will need further actions if we provide the results in Figure 1b (high relevance) in the first page. A higher diversified result set (like the one in Figure 1d) would have been more appropriate. If instead, a more specific query is posed, like DSLR Camera, then higher relevance and lower diversity is preferable, because the user may (with high probability) satisfy her search with just one page. Note that this dynamic balancing of relevance versus diversity can also occur, within the subsequent navigation steps of the same query, as it is progressively refined by the user (e.g. after posing the Camera query and getting the results in Figure 1d, a user interested in lens might refine by selecting the condition Type: Lens).

To this end, we propose a user navigation model that considers factors such as the characteristics of the query result, the user's familiarity with various refine conditions, the number of pages the user would have to navigate and the expected number of navigation steps required to reach a result of interest. The resulting model is adaptive to user actions and constructs a diversified result that minimizes the expected user effort. This is in contrast to the fixed diversity vs. relevance trade-off achieved by previous techniques, which leads to a much higher navigation cost, as shown in our experiments. In summary, we make the following contributions:

1. A user navigation cost model that captures the actions of a user navigating a result set (Section 2).
2. The cost model is necessarily probabilistic since it computes the cost based on possible future actions taken by the user while navigating a result set. We propose ways of estimating the expected cost in Section 3.
3. We show that the problem of computing the best set of results to minimize the expected user effort is NP-hard and propose efficient approximate algorithms to compute an appropriately balanced diversified result-set that minimizes the expected user navigation cost (Section 4).
4. We present the results of an extensive experimental evaluation of the proposed techniques compared to state-of-the-art ranking methods using two real datasets (Section 5).
5. We validate our cost model and measure user navigation time with a user study at Amazon Mechanical Turk, which

shows that our methods outperform the state-of-the-art (Section 6).

Related work is discussed in Section 7 and we conclude in Section 8.

2. PROBLEM DEFINITION

We proceed with the problem setting (Section 2.1), and describe our navigation cost model (Section 2.2), which mimics the actions of a user navigating query results and quantifies the user effort, that is, assigns cost to the user actions. We then conclude with a formal problem definition (Section 2.3).

2.1 Preliminaries

Database: Let $D = \{r_1, \dots, r_n\}$ be a database consisting of n tuples and $\mathcal{A} = \{A_1, \dots, A_m\}$ be a set of attributes. Each attribute has an associated domain $ADom(A_i)$ consisting of un-interpreted constants. The database D is heterogeneous and each tuple $r_i \in D$ has a value for a subset $A^i \subseteq \mathcal{A}$ of attributes and a *null* (ϵ) value for the rest.

Query: The user exploring D, navigates the results $R_Q \subseteq D$ of a query Q which could be a keyword query. Each attribute-value combination in the results of R_Q, denoted by $c: A_i = v_i$, is a *condition* and can be used to *refine* the query. We denote the set of all conditions of R_Q by $C(R_Q)$.

Example: Figure 1a shows a result-set R_Q for *Q: Camera*, where the user could refine the query by selecting condition c_i : *Brand = Canon,* which would return the two Canon cameras (#1, #3).

Paginated Result Subset S_k: Typically, the result set R_Q is too large to fit into a single page on the user interface. These results are therefore *paginated* and only a small subset $S_k \subseteq R_Q$ of size k is presented to the user at a time. The size k of the result subset depends on the display size of the device. For example, e-commerce and web-search interfaces typically show 10-15 results at a time on desktop browsers and 5 on mobile devices. The latter are the key focus of this paper, which are even more challenging because the screen size does not allow displaying other result information like facets (see discussion in Section 7).

Example: Figures 1b-d show examples of various paginated result subsets (with $k = 3$) for the query results in Figure 1a.

The choice of the subset S_k is critical in determining the effectiveness of a search interface. In particular, S_k should contain relevant and a diverse set of results.

Let $rel(r, Q)$ be the relevance score of a result $r \in R_Q$, where a higher score means the result r is more relevant to query Q. Existing object relevance ranking functions like [8] can be used to compute $rel(r, Q)$. Also, let $dist(r_i, r_j)$ be the distance between two results $r_i, r_j \in R_Q$. For instance, $dist(r_i, r_j)$ can be the Euclidean distance between the vectors representing r_i and r_j. Currently, most systems model search result diversification as a bi-criteria optimization problem that balances the effect of relevance and diversity using a trade-off parameter ($\lambda \in [0,1]$) as follows:

$$S_k = argmax_{S \subseteq R_Q, |S|=k} \left(\begin{array}{c} (k-1)(1-\lambda) \sum_{r_i \in S} rel(r_i, Q) + \\ 2\lambda \sum_{r_i, r_j \in S} dist(r_i, r_j) \end{array} \right) \quad (1)$$

In Section 1, we described several scenarios where this bi-criteria optimization is problematic. The primary reason is the difficulty in selecting a value of λ for a given navigation step of a query.

Instead of fixing the trade-off between relevance and diversity, we model the diversification problem in terms of the user navigation effort. We do this by designing a holistic model of the user navigating a list of paginated results that considers all the actions taken by a user, discussed next.

2.2 Navigation Cost Model

Users execute queries on a search interface to satisfy a certain information need, which may be satisfied by a certain objects in the query result. Given a user query Q and corresponding result set R_Q, the query interface presents the first page of results $S_k \subseteq R_Q$ of size k to the user. Each result consists of a set of attribute-value conditions. These conditions are *selectable* (e.g. by clicking on the associated link), thereby refining the query. Note that the search interface does not provide any facet conditions to refine the results, but the results serve this dual purpose. Such an arrangement is desirable especially in mobile devices where there is not enough space to show both results and facet conditions.

NAVIGATE (Q, R_Q)
1. S_k = COMPUTE-PAGE(Q, R_Q)
2. READ-RESULT(S_k)
3. If search need satisfied by S_k
4. TERMINATE
5. Choose one of the following:
6. (a): Select a condition $c \in C(S_k)$ to refine; REFINE(Q, c): $Q \leftarrow Q \wedge c$
   (b): NEXT-PAGE(R_Q, S_k): $R_Q \leftarrow R_Q \backslash S_k$
7. NAVIGATE(Q, R_Q)

Figure 3: Diversification Navigation Model

In particular, the user chooses among the following possible actions at each step:

1. $TERMINATE$: If the user's search need is satisfied on the current page, the search is terminated.
2. $NEXT-PAGE(R_Q, S_k)$: The user can navigate to the next page of the result set in the hope of satisfying her search need there. In this case, the search interface computes the next page of results and presents these results to the user.
3. $REFINE(Q, c)$: Typically, a user has a notion of the properties or conditions that a target (desired) object must have. If one of these conditions is found in one of the displayed results, then the user can refine her query by selecting (clicking) this condition *c*. The query is then refined to $Q \wedge c$. For example, if a user is looking for *Compact* cameras while reading through the result subset in Figure 1d, she could click on the *Compact* attribute value. This user repeatedly executes REFINE and NEXT-PAGE actions until the target object is found, at which point the user TERMINATEs the navigation. This iterative result navigation process is captured by the recursive navigation model presented in Figure 3. At the beginning of each step (line 1), the system computes a page of k results to be presented to the user. The user reads all the results, represented by READ-RESULT(S_k) and the rest of the navigation repeats recursively.

In our model, the user executes an action based on the displayed result set. Such navigation models, where the user only selects

actions proposed by the system, have been used extensively in the navigation of keyword-based query results [12][21][22]. Each user action (REFINE, READ-RESULT, NEXT-PAGE) is an effort on the part of the user. The total effort of the user to satisfy her search need is the *navigation cost*.

As an example, consider the user navigating the result set in Figure 1a using the initial pagination in Figure 1d. Further assume that the user is interested in *Lenses* with *55-300 mm Focal Length* (#11). As a first step, the user would read (READ-RESULT) the first page of 3 results. Next, the user REFINEs by *Type: Lens* to see only the *Lens* results, since she is interested in *Lenses*. Up to this point the navigation cost consists of 3 READ-RESULT actions and 1 REFINE. Upon REFINE, the interface presents the 2 lens results (#10,#11), which the user reads and finds the desired object, thereby TERMINATING the navigation. The overall navigation cost is $5 \cdot$ READ-RESULT + $1 \cdot$ REFINE.

If we assume that reading a result incurs unit cost (as was assumed in [12]), and the cost of REFINE (click) action is a constant greater than one (say $\alpha = 3$), the total cost is $5 \cdot 1 + 1 \cdot 3 = 8$. This assumption about the constant α reflects our belief that REFINE incurs more user effort than reading a result, since the user has to consider all the conditions and then decide on a condition to click on. Similarly, the user could do NEXT-PAGE instead of REFINE if she does not find any useful condition to refine on. The cost of NEXT-PAGE is β.

2.3 Problem Statement

The overall navigation cost depends on the result subset S_k that is presented to the user at each step. For example, if the user in the example above is presented with a paginated result subset containing the Camera *Lens* with *55-300 Focal Length* (#11), then she would find the desired *target* object on the first page and TERMINATE the search. In this case, the total navigation cost is 3 (3 READ-RESULTs).

Therefore, we need to compute a paginated result subset $S_k \subseteq R_Q$, that will minimize the expected navigation cost, by appropriately balancing relevance and diversity. Let $cost(Q, R_Q, S_k)$ denote the cost of navigating the result set R_Q of a query Q, using the paginated result subset S_k. Then the minimal cost of navigation, $cost(Q, R_Q, k)$, is the cost of the paginated result set S_k^{opt}, amongst the $\binom{|R_Q|}{k}$ k-subsets, that has the minimum cost. Formally,

Problem 1: Minimum Cost Diversification(R_Q, Q, k): Given a query Q and its result set $R_Q (|R_Q| \geq k)$, compute the result subset S_k^{opt} of size k $\left(|S_k^{opt}| = k\right)$ such that the expected navigation cost incurred to satisfy the user's search need is minimized.

$$cost(Q, R_Q, k) = min_{S_k \subseteq R_Q} cost(Q, R_Q, S_k) \quad (2)$$

Next, we show how to estimate $cost(Q, R_Q, S_k)$, the cost of a result subset S_k for a result set R_Q.

3. NAVIGATION COST ESTIMATION

The navigation cost of a result set R_Q, computed as discussed in Section 2.2, depends on the actions taken by the user in reaching a target object and can be exactly determined after the navigation is complete. However, solution to the minimum cost diversification problem (Problem 1) requires the selection of result subset S_k before knowing what sequence of actions the user will perform after viewing S_k. In this section, we propose a way to estimate the cost of navigating a result subset S_k by means of a probabilistic cost model that assigns uncertainty measures to each possible action a user can take and computes the expected navigation cost for a given S_k.

We begin by introducing the probability measures that capture the uncertainty in user action. In the user navigation model in Figure 3, the user, at each navigation step, has three choices – (1) TERMINATE the navigation (line 3) (2) REFINE by a condition (line 6a) and (3) to go to the NEXT-PAGE of results (line 6b), and we introduce probability measures for each of these actions.

- **P_T (*TERMINATE Probability*):** This is the probability that the user finds the result she is looking for in S_k, and therefore terminates the navigation process.
- **P_R (*REFINE Probability*):** This is the probability that the user chooses to refine the result set R_Q by adding a condition c to the query Q. On the other hand, the user could instead choose to see the next page of the results. Since these are the only two choices supported by the navigation model, the probability that the user chooses the NEXT-PAGE action is $1 - P_R$.
- **P_c (*REFINE* by condition *c Probability*):** If the user chooses to REFINE, then she also has to select a condition $c \in C(S_k)$ to refine by. The probability P_c captures the probability that the user selects a condition c.

Given the probabilities defined above, the entire navigation process, which contains several iterations, can be expressed by the following recursive cost equation (*cost(.,.,.)* is overloaded):

$$cost(Q, R_Q, S_k) = |S_k| + (1 - P_T) \cdot \left[P_R \left\{ \alpha + \sum_{c \in C(S_k)} P_c \, cost(Q \wedge c, R_{Q \wedge c}, k) \right\} + (1 - P_R)\left(\beta + cost(Q, R_Q \setminus S_k, k)\right) \right] \quad (3)$$

This cost equation can be described as follows:

1. The user reads the results in S_k, and decides about her next action. The cost for reading the results is $|S_k|$ (assuming unit cost for the READ-RESULT action). If the user finds the target object then she terminates the navigation.
2. Otherwise, with probability $(1 - P_T)$, she can either refine the query or go to the next page,
   a. The user decides to refine the query with probability P_R. Let α be the cost for a REFINE action. As the user can select any condition $c \in C(S_k)$,we consider the cost associated with each selection candidate c (shown as $cost(Q \wedge c, R_{Q \wedge c}, k)$) weighted by the P_c value.
   b. With probability P_N the user decides to go to the next page. β is the cost of a NEXT-PAGE action, and the cost entailed by the NEXT-PAGE action and the cost of the rest navigation is $cost(Q, R_Q \setminus S_k, k)$.

The cost equation (Equation 3) depends on the key probability terms P_T, P_R and P_c. We present specific and reasonable methods to compute the probabilities used in our cost model. Depending on the specific application, other computation methods may more closely model the user. The computation of these probabilities is orthogonal to the algorithms presented in Section 4, which are the key contributions of this paper.

Computing P_T: This is the probability that the user finds the target object in S_k and therefore terminates the navigation. Since

the target object is not known before navigation, a reasonable assumption is the probability of a potential target object is proportional to its *relevance* score $rel(r, Q)$. If the user finds the target object amongst the result subset S_k, then she can terminate the navigation. Therefore, we estimate the probability of termination as being proportional to the sum of relevance scores in the paginated result subset S_k and normalize it with sum of relevance scores of all the results in R_Q as $P_T = \sum_{r \in S_k} rel(r, Q) / \sum_{r \in R_Q} rel(r, Q)$. The choice of the relevance function is orthogonal to this paper and can be computed in various ways such as TF-IDF [13] for keyword queries or by using Probabilistic Information Retrieval models [8].

Computing P_R : The key assumption we make is that the user has a high likelihood of refining the query when the results in S_k are diverse. This is because a diverse S_k covers multiple aspects of the query by containing results from different interpretations [7], [24]. For example, the result subset in Figure 1d contains results from different categories and also contains a variety of attribute values, and hence offers many refinement opportunities, which translated to a high P_R. In contrast, the result subset of Figure 1b would translate to a low P_R.

To compute the diversity of a result subset S_k, we need to compute the distance between all pairs of results $r_i, r_j \in S_k$ [5]. Distance measures like Euclidean distance, Cosine Similarity can be used for this purpose and again the choice is orthogonal to this paper. The diversity of S_k can be defined as $div(S_k) = \sum_{r_i, r_j \in S_k} dist(r_i, r_j)$. Hence, P_R can be computed as $P_R = \frac{\sum_{r_i, r_j \in S_k} dist(r_i, r_j)}{max_{S \subseteq R_Q, |S|=k} \sum_{r_i, r_j \in S} dist(r_i, r_j)}$, where the denominator is used for normalization and is equal to the maximum possible diversity of a k-result set from R_Q. Finding the maximum diversity in the denominator is similar to the maximum clique problem, and therefore known to be NP-Complete [5]. We use MMR [10] to compute the maximum diversity for our experiments (in Section 5).

Computing P_c: This probability shows how likely the user will refine the query by selecting condition $c \in S_k$, which depends on the user's preferences or her familiarity to this condition. A reasonable choice is that P_c is proportional to the frequency of condition c in the result set as described in [12].

4. ADAPTIVE DIVERSIFICATION

Exact Algorithm: To compute the paginated result set such that cost of navigating R_Q is minimized, it is necessary to compute the cost, using Equation 3, of each subset $S_k \subseteq R_Q$ of size k and selecting the subset S_k^{opt} that has the minimum cost. We show that this problem (Problem 1) is NP-hard [28], by reducing Set Cover to a simplified version of this problem. There are two sources of complexity that make the exact algorithm computationally expensive:

1. Computing the navigation cost of each subset $S_k \subseteq R_Q$ of size k requires evaluating Equation 3 for $O\left(|R_Q|^k\right)$ subsets.
2. Since Equation 3 is recursive, to solve it we must compute for each condition c in S_k (more formally $c \in C(S_k)$), the minimum cost (according to Equation 2), which in turn requires computing the minimum cost over all subsets of R_Q (Figure 4(a)). This process continues recursively for deeper levels of the recursion tree. The depth of the recursion is $|R_Q|$ in the worst case, since each refinement may eliminate just one result in R_Q. The width of each recursive step, i.e., the cardinality of the summation, can be up to $m \cdot k$, which is the number of attribute values displayed at each step.

Approach overview: We attack the problem by proposing efficient techniques to approximate both sources of complexity. We first show how to effectively eliminate the recursion from Equation 3 using a sequence of two relaxations. Then we show how to avoid evaluating the simplified equation for every combination of result subsets using a greedy algorithm.

Our approach starts by eliminating recursion from Equation 3 using two relaxations. Figure 4 shows the recursive tree to compute Equation 3 and the simplifications achieved through the two relaxation steps.

Relaxation 1 (Eliminate Conditions from Recursion Tree): The navigation cost function (Equation 3) has two recursive calls – one each for REFINE and NEXT-PAGE actions, respectively to compute the navigation cost for subsequent navigation steps.

Intuitively, the navigation cost of a result-set R_Q is proportional to its size $|R_Q|$, since for a larger result-set the user must explore more results to reach the results of interest. This assumption is backed by our experiments in Section 5 (specifically Figure 6) and we use this observation to simplify the cost equation. Formally,

$$cost(\mathrm{Q}, R_Q, k) \propto |R_Q| \qquad (o1)$$

The cost associated with REFINE actions, denoted by $cost(Q \wedge c, R_{Q \wedge c}, k)$, is the navigation cost incurred to reach the target results from the refined result set $R_{Q \wedge c}$. Based on the observation above, this cost is proportional to size of $R_{Q \wedge c}$, i.e.

$$cost(Q \wedge c, R_{Q \wedge c}, k) \propto |R_{Q \wedge c}| \qquad (o2)$$

Based on observations $o1$ and $o2$ and ignoring the constants of proportionality, the cost of the REFINE by a condition c can be estimated as:

$$cost(Q \wedge c, R_{Q \wedge c}, k) = \left(|R_{Q \wedge c}| / |R_Q|\right) cost(\mathrm{Q}, R_Q, k)$$

Analogously, the cost of NEXT-PAGE action $\left(cost(Q, R_Q \setminus S_k, k)\right)$ can be approximated as $\frac{|R_Q \setminus S_k|}{|R_Q|} cost(\mathrm{Q}, R_Q, k)$, since the user is left with $R_Q \setminus S_k$ of the original result-set R_Q after a NEXT_PAGE action.

By plugging in these approximations and rearranging terms, our cost equation can be rewritten as:

$$cost(\mathrm{Q}, R_Q, S_k) = |S_k| + (1 - P_T) \cdot \left\{ \begin{array}{c} \alpha P_R + \beta(1 - P_R) + \\ cost(\mathrm{Q}, R_Q, k) \left[P_R \sum_{c \in C(S_k)} P_c \frac{|R_{Q \wedge c}|}{|R_Q|} + (1 - P_R) \frac{|R_Q \setminus S_k|}{|R_Q|} \right] \end{array} \right\} \quad (4)$$

Equation 4 replaces the recursive calls of REFINE and NEXT-PAGE actions (in Equation 3) with $cost(\mathrm{Q}, R_Q, k)$(Figure 4(b)). However, it still requires evaluation of all possible *k*-subsets of R_Q to compute $cost(\mathrm{Q}, R_Q, k)$ according to Equation 2. We address this by the next relaxation.

Relaxation 2 (Eliminate Result Subsets from Recursion Tree): Our goal is to find the result subset S_k that minimizes Equation 4. But note that this same optimal S_k is used to

compute $cost(\mathrm{Q}, R_Q, k)$ according to Equation 2. Hence, we can replace $cost(\mathrm{Q}, R_Q, k)$ by $cost(\mathrm{Q}, R_Q, S_k)$ in Equation 4.

Then, by solving for $cost(\mathrm{Q}, R_Q, S_k)$, Equation 4 can be further simplified as:

$$cost(\mathrm{Q}, R_Q, S_k) = \frac{|S_k| + (1 - P_T).\{P_R.\alpha + P_N.\beta\}}{1 - (1 - P_T).\{P_R.\sum_{c \in C(S_k)} P_c.\frac{|R_{Q \wedge c}|}{|R_Q|} + P_N.\frac{|R_Q \setminus S_k|}{|R_Q|}\}} \quad (5)$$

Equation 5 has no recursion, and can be easily computed for a given S_k. Note that Relaxation 2 does not incur any approximation error, in contrast to Relaxation 1.

Given the relaxed cost Equation 5, we still need to compute the cost of all possible k-result subsets S_k of R_Q to find the optimal S_k^{opt} with minimum cost.

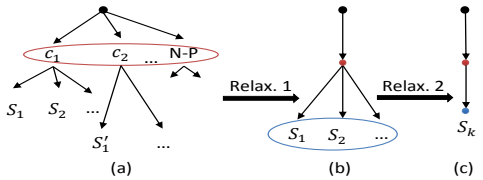

Figure 4: Elimination of Recursion Using Relaxations 1&2

Algorithm: $ADA(Q, R_Q)$
Input: Query Q and Result Set R_Q
Output: Set of k results, $S_k \subseteq R_Q$

1. $S_0 \leftarrow \emptyset$
2. $E \leftarrow \mathrm{R_Q}$
3. $r \leftarrow argmax_{r_i \in R_Q} rel(r_i, Q)$
4. $S_1 \leftarrow S_0 \cup r$
5. $E \leftarrow E \setminus \mathrm{r}$
6. for p $\leftarrow$ 2 to k do
7. $r \leftarrow argmin_{r_i \in E} cost(\mathrm{Q}, R_Q, S_{p-1} \cup r)$
8. $S_p \leftarrow S_{p-1} \cup r$
9. $E \leftarrow E \setminus \mathrm{r}$
10. return S_k

Figure 5: Adaptive Diversification Algorithm

For that, we present an efficient greedy algorithm, called *Adaptive Diversification Algorithm (ADA)*, which incrementally builds the result set S_k by adding at each step the result with minimum incremental navigation cost. At each iteration p ($0 \leq p \leq k$), ADA makes use of two sets: the set of remaining results E and the set of selected results S_p, with $|S_p| = p$. Note that $E \cup S_p = R_Q$, the set of all the results. Initially $E = \mathrm{R_Q}$ and $S_0 = \emptyset$. At each iteration, ADA computes $cost(\mathrm{Q}, R_Q, S_{p-1} \cup r)$ (using Equation 5) for each result $r \in E$ and moves the result with minimum navigation cost to S_p. This process continues until we select k results (i.e. $p = k$).

Figure 5 shows the pseudo-code of our diversification algorithm ADA. The first result is chosen as the object with highest relevance score (line 3) since we want to provide to the user the most relevant object. After that, in each iteration, ADA ranks the results in E according to Equation 5 with S_k replaced by $S_{p-1} \cup r$, removes the result with minimum navigation cost from E, and adds it in the selected result set (line 6-9). The algorithm terminates when we select k results.

Complexity: The running time of ADA depends on the cost computation time in line 7, which is invoked up to $O(k \cdot |\mathrm{R_Q}|)$ times. To compute $cost(\mathrm{Q}, R_Q, S_{p-1} \cup r)$ using Equation 5, we need to calculate P_T, P_R, P_N, the cost of all possible REFINEments ($\sum_{c \in C(S_k)} P_c \cdot \frac{|R_{Q \wedge c}|}{|R_Q|}$) and of NEXT-PAGE action ($\frac{|R_Q \setminus S_k|}{|R_Q|}$). Computation of P_T and P_R (Section 3) is dominated by their denominators, which depend on the result set $\mathrm{R_Q}$ and k. However, in ADA we only need to compute these probabilities for the original result-set $\mathrm{R_Q}$, which takes time $O(|\mathrm{R_Q}|)$ and $O(k \cdot |\mathrm{R_Q}|)$(using MMR [10]), respectively. The computation of all REFINEments cost ($\sum_{c \in C(S_k)} P_c \cdot \frac{|R_{Q \wedge c}|}{|R_Q|}$) requires $O(k \cdot m + |\mathrm{R_Q}|)$ time. The cost of NEXT-PAGE and P_N can be computed in $O(1)$ time. Therefore, the total running time of ADA is $O(k \cdot |\mathrm{R_Q}|^2)$ (assuming $|\mathrm{R_Q}| > k.m$). In practice, the execution time is much faster than this worst case bound (Section 5).

Example: Let us apply ADA to the result set in Figure 1a. We are interested to find the 3 results returned by ADA. But before that we analyze Equation 5 more closely to infer the implication of the cost equation. The cost of a result set S_k is minimized when the denominator of the right hand side in Equation 5 is maximized, which implies having higher P_T value. But when the result set size $|\mathrm{R_Q}|$ is high, we have smaller P_T value (since the denominator part of P_T in Section 3 is high). Therefore, the navigation cost depends on the cost of REFINE and NEXT-PAGE actions. Since the NEXT-PAGE cost (denoted as $\frac{|R_Q \setminus S_k|}{|R_Q|}$) is the same for all result sets, the navigation cost is minimized for the result set S_k containing highly diverse results with popular selective conditions (i.e., with high P_c). As $|\mathrm{R_Q}|$ becomes smaller, P_T dominates cost equation, therefore cost is minimized for highly relevant results.

The conclusion of the above discussion is that, initially when $|R_Q|$ is large (small P_T), ADA prefers diversity over relevance. As $|R_Q|$ becomes smaller (higher P_T) in the next iterations, ADA increases preference to relevance, and provides highly relevant results.

Returning to the running example, ADA initially prefers diversity over relevance in Figure 1a since $|R_Q|$ is relatively large. The first result is the result #1 as it has the highest relevance score (diversity is not a factor when selecting the first result, which is always selected by relevance). The second result would be from *Compact* or *Accessory* categories. Assuming all the conditions in *Compact* and *Accessory* categories have similar selectivity (similar P_c and similar diversity with respect to #1), the second result is #6 because of its high relevance score. The third result would be from the *Accessory* category to increase diversity, and specifically #10 since it has higher relevance score than #11. Thus ADA would return result set in Figure 1d. In the next iteration, as R_Q becomes smaller, ADA will return more relevant result-set snippets like the one in Figure 1c, and in the last iterations like the ones in Figure 1b.

5. EXPERIMENTAL EVALUATION

In this section, we describe the results of an extensive experimental evaluation of our approach. The setup, including methodology, datasets, baselines and metrics used, is described in Section 5.1. Sections 5.2 and 5.3 demonstrate the quality and performance of diversification algorithms in terms of user

navigation cost and CPU time. All experiments were performed on a 2.5GHz Intel Core i5 CPU, 8GB RAM machine running Windows 7. We used MySQL as our database and all algorithms were implemented in Java.

5.1 Setup

Datasets: We evaluated our approach on two datasets:

1. **UsedCars:** This dataset consists of a listing of 15,191 used cars, extracted from a popular car-trade website. Each tuple has 10 attributes, 4 categorical and the rest numeric.
2. **Electronics:** This dataset consists of 65K products from the *Electronics* product catalog of a popular e-commerce website. The products were sampled from various *Electronics* categories, such as *Laptops, Desktops, Cameras, Printers* etc. and therefore the dataset is highly heterogeneous in nature. The dataset has a total of 86 (51 categorical and 35 numeric) attributes, but each product has values for a small subset (avg. 12) of these attributes and *null* for the rest.

Queries: We selected 8 queries each from the two datasets. These queries are shown in Table 1 along with result-set sizes. Note that we are interested optimizing the navigation of diverse result sets, and therefore these queries were selected to be deliberately ambiguous so as to include results from a variety of categories. For that, we use single-keyword queries, although our methods support any number of keywords or query conditions; more keywords could be used if larger e-commerce datasets were available. For each query, we select a target object, which we assume the user is looking for, i.e. the navigation terminates when the user locates this target object.

Table 1: Query Set

Electronics			UsedCars		
Query ID	Query	# Results	Query ID	Query	# Results
Q_1	Kodak	193	Q_9	Honda	789
Q_2	Dell	125	Q_{10}	BMW	730
Q_3	Canon	1097	Q_{11}	2001	2034
Q_4	Nikon	511	Q_{12}	2005	920
Q_5	Camcorder	789	Q_{13}	Dallas	2932
Q_6	Speaker	737	Q_{14}	Irving	1064
Q_7	Desktop	394	Q_{15}	Black	2163
Q_8	Laptop	518	Q_{16}	Blue	1183

State-of-the-art: Current approaches to diversification use a fixed *relevance-vs.-diversity* trade-off parameter (λ in Equation 1) to diversify rankings. However, as we argued in Section 1, setting this parameter is not always obvious and depends on the characteristics of the result set. In Section 5.2, we provide evidence to further support this claim. We compare with two commonly used approaches for ranking results:

1. **Baseline 1 (REL):** In this approach, the results were ranked solely based on relevance, i.e. by setting $\lambda = 0$ in Equation 1.
2. **Baseline 2 (MMR-$\lambda = 0.5$):** As a second baseline, we choose the Maximal Marginal Relevance (*MMR*) diversification algorithm [10]. *MMR* computes a diversified result-set by balancing relevance and diversity based on Equation 1. *MMR* is an approximation algorithm since computing a diversified set based on Equation 1 is NP-Complete [5]. In our experiments, we set $\lambda = 0.5$ giving equal weight to diversity and relevance factors.

Note that, for a fair comparison, we used *MMR* both as a baseline and to compute P_R in *ADA* algorithm. We chose *MMR* since it outperforms other algorithms in terms of time and generates quality results [5]. However, our use of *MMR* does not preclude the use of other diversification algorithms, e.g. GMC, GNE [5], which may produce better quality results but take more time compared to *MMR* [10].

Next, we describe the relevance (rel) and diversity $(dist)$ measures used in our experimental evaluation. We reemphasize that computing these measures is orthogonal to our problem and any suitable rel and $dist$ versions can be plugged in to our approach. Due to space limitations we omit experiments with additional measures.

Computing $dist$: Computing diversity involves computing distance $(dist)$ between two results. We use the sum of distances between the attribute values $dist(r_i, r_j) = \sqrt{\sum_{A_k \in \mathcal{A}} \left(r_i(A_k) - r_j(A_k)\right)^2}$. For numeric attributes, we used the Manhattan distance and for categorical attributes, the Kronecker delta function was used between the values of attribute.

Computing rel: In a structured result-set, the relevance of a result depends on relevance of its attribute values. We estimated the relevance of each attribute value by computing the Google Trends scores (see [8] for more details). The rationale for using Google Trends is based on the idea that the relevance of a term can be based on its frequency in a query workload [8].

Since results in R_Q satisfy all conditions in Q, the relevance scorewas computed using the unspecified attributes in $\mathcal{A}$ by Q, as was proposed by [8], where unspecified refers to an attribute that does not match any query condition. For example in Figure 1a, all the records satisfy the query condition *"Camera"* with their *Product* attribute. Therefore, we compute the relevance score using the unspecified attributes (e.g. *Category, Brand*).

Methodology: For each query in Table 1, we picked a result $t \in R_Q$ as the target object. The chance of selecting a result as the target object is proportional to its relevance score, which means the results with high relevance scores have a higher chance to be selected as the target object. We then simulated the user navigation until target t is reached. Since multiple navigation paths can lead to the target object t, we used a *randomized simulation* [12] to select navigation paths. Note that, given a set of displayed results S_k, the set of conditions that can lead to t is $C(S_k) \cap C(t)$. We assumed that the user will select one of these conditions, or go to next page, according to the navigation probabilities (Section 3). For example in Figure 1a, if the target object is #4, and we select Figure 1c as the displayed result subset, the conditions that lead to #4 are "Product=Camera", "Category = DSLR" and "Brand = Nikon". The user would go to the next page if she does not like or know these three conditions. Therefore, in our simulation, we computed P_N as $\prod_{c \in C(S_k) \cap C(t)} (1 - P_c)$ (the probability that the user would not like any condition in $C(S_k) \cap C(t)$) and P_R as $(1 - P_N)$. In case of refinement, the user could refine the query by selecting any condition in $C(S_k) \cap C(t)$. The choice of selecting a condition $c \in C(S_k) \cap C(t)$ is proportional to P_c.

We used $k = 10$ in the experiments in Section 5.2, 5.3, and showed the findings averaged over 1000 runs (50 random target objects, and 20 runs per target object) for each query.

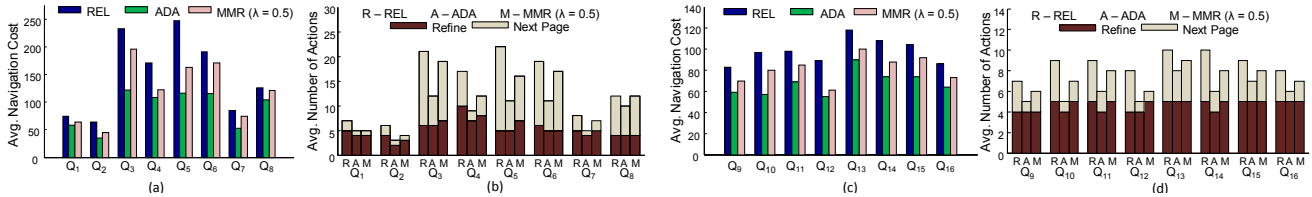

Figure 6: (a) Avg. Navigation Cost (b) Avg. Number of Refine and Next Page actions incurred for Electronics Dataset using α=1, β=1. (c), (d) show the same figures respectively for UsedCars Dataset

5.2 Qualitative Analysis

In this section, we present the experimental results of the qualitative evaluation of the three different algorithms (*REL*, *MMR* and our algorithm *ADA*). Figures 6(a), 6(b) show the average navigation cost and average number of *REFINE* and *NEXT PAGE* actions incurred respectively by each algorithm to reach the target object for the queries of *Electronics* dataset. Note that, all algorithms require similar number of *REFINE* actions (i.e. selection of target object conditions) to filter out enough undesired objects (Figure 6b). Since *REL* displays results from popular categories, it requires a larger number of *NEXT PAGE* actions to display the conditions of the less relevant target objects. *MMR* has a fixed ratio of relevance and diversity, which happens to work well for some queries with small number of results like Q_1, Q_2 and Q_4, but is ineffective for other queries like Q_3, Q_6, where more *NEXT PAGE* actions are required to find the target object conditions.

ADA outperforms the other two algorithms, because of its adaptive nature. As discussed in Section 4, when $|R_Q|$ is high, *ADA* prefers diversity over relevance to pick the Top-k results. As R_Q becomes more selective over iterations, *ADA* switches to preferring relevance. Therefore, by balancing diversity and relevance based on the result set at hand, ADA displays the target object conditions much earlier compared to the other two algorithms. This results in fewer *NEXT PAGE* actions, and thus reduces the navigation cost of *ADA* algorithm (Figure 6a). The improvement of *ADA* over the other two algorithms is more pronounced for the queries that have large number of results (e.g. Q_3, Q_5, Q_6).

Figures 6(c), 6(d) show the average cost and actions respectively for the queries of *UsedCars* dataset. Similar to the *Electronics* dataset, *ADA* outperforms the other two algorithms for all the queries of *UsedCars*. Since the *UsedCars* dataset is homogeneous, *REL* and *MMR* perform slightly better as compared to *Electronics* dataset, due to less variability in attribute conditions.

Table 2: Average Navigation Cost for the Datasets

Datasets	Average Navigation Cost (α=1, β=1)			
	REL	ADA	MMR (λ = 0.5)	Expected Optimal
Electronics	48.25	30.625	38.75	28.525
UsedCars	30.5	19.15	24.375	18.512

We also compare the average navigation cost incurred by the three algorithms, *REL*, *MMR* and *ADA*, with the expected *optimal* navigation cost computed by solving Equations 2 and 3. Due to the exponential complexity, we compute the expected optimal cost for smaller sizes of initial result sets (R_Q) and query parameter (k). Table 2 shows the average navigation costs for $|R_Q|$ = 100 and k = 5 across all queries for the two datasets *Electronics* and *Usedcars*. We see that our algorithm *ADA* is only 1.07 and 1.03 times worse than the optimal for the *Electronics* and *Usedcars* datasets respectively. For *MMR* and *REL*, these factors are 1.36 (1.32) and 1.69 (1.65) respectively for the *Electronics* (*Usedcars*) dataset.

Figure 7 shows average navigation cost of *MMR* with increasing trade-off (λ) values (high λ value implies preference to diversity over relevance) for four queries from Table 1. The results of the other queries are shown in [28]. Since *ADA* is independent of λ value, therefore the cost of *ADA* is shown as a straight line. We skipped *REL* since it incurs higher cost compared to *ADA* and *MMR*. As seen from Figure 7, there is no value for λ that is optimal for a given dataset or even for a particular query. Intuitively λ should change adaptively, at each navigation step, depending on the characteristics of the result set. By balancing the relative importance of relevance and diversity adaptively at each step, *ADA* shows better performance (on average) compared to *MMR* with a fixed λ.

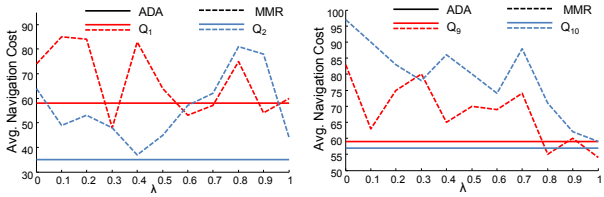

Figure 7: Average Navigation Cost vs. Tradeoff (λ) values

We also perform experiments while varying the model parameters (α, β). The results depict that the overall cost of *ADA* increases at a slower rate with the model parameters compared to *REL* and *MMR* [28].

5.3 Performance Analysis

We now present the performance results of *REL*, *MMR* and our *ADA* algorithms. Table 3 shows the average (across all queries) *computation* (CPU) times taken by each of these algorithms to reach the target object. As expected, the relevance-only algorithm (*REL*) takes the shortest time among all the algorithms. Computing diversity is a costly operation since it involves computation of distance between all pairs of results. As a result any algorithm that incorporates diversity is much slower as compared to *REL*; our implementation of *MMR* is *three times* slower as *REL*. Our algorithm (*ADA*) performs this distance computation over all future navigations and therefore is slower than *MMR* by a factor of 1.6. While *ADA* takes more time to compute the set of paginated results, it is very effective in reducing the time or effort incurred by users to navigate such diverse result sets (discussed in Section 6).

Table 3: Avg. CPU Time for the Datasets

Datasets	CPU Time (sec)		
	REL	ADA	MMR (λ = 0.5)
Electronics	0.02	0.106	0.066
UsedCars	0.058	0.156	0.107

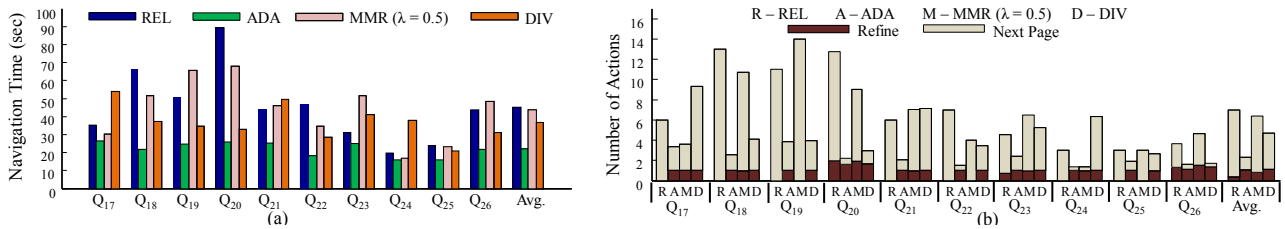

Figure 8. (a) Avg. User Navigation Time, (b) Avg. number of Refine and Next Page Actions incurred by the users for 10 different queries using UsedCars dataset (for α=1, β=1)

We also examine the scalability of our algorithm (*ADA*) by applying it to large result set and find that our algorithm scales almost linearly with result set size [28].

6. USER STUDY

In this section, we present the results of a user study that we conducted at Amazon Mechanical Turk (*MTurk*) [26] using the *UsedCars* dataset. We selected three keyword queries (e.g., *'Ford'*), and for each query we created a set of search tasks; each search task specifies a set of target conditions (e.g., find a car with Color = Green). We asked the users, starting from the results of the initial keyword query, to find the best car (according to the relevance score defined in Section 5.1) that satisfies all the target conditions.

Table 4: Query Set for User Study

Query ID	Initial Query	Initial Result Set Size	Target Conditions	# of Results contain all Target Conditions
Q_{17}	Toyota	1470	Color = Green	86
Q_{18}	Ford	2747	City = Grand Prairie	133
Q_{19}	Ford	2747	Model = F150 Regular CAB, State = MD, Color = Maroon	1
Q_{20}	Ford	2747	Color = Maroon	42
Q_{21}	Ford	2747	City = Ashland	75
Q_{22}	Ford	2747	Color = Beige	40
Q_{23}	Toyota	1470	City = Richmond	88
Q_{24}	Toyota	1470	Color = Red;	79
Q_{25}	Ford	2747	Color = Gold;	116
Q_{26}	BMW	730	Model = Convertible, City = Fairfax, Color = Grey;	1

We repeated the experiment for the four different ranking algorithms: *REL* ($\lambda = 0$), *MMR* ($\lambda = 0.5$), *ADA* (λ-independent) and a diversity-only baseline, *DIV*, which constructs the *k*-result subset greedily at each step by maximizing the score function (Equation 1); i.e. with $\lambda = 1$. The reason that we asked users to find the best and not any result is to avoid giving an unfair advantage to methods biased towards diversity like *DIV* which may help the user to find a satisfying result, but this result may have low relevance. Table 4 shows the list of initial queries, their results' cardinality, the target conditions, and the cardinality of results that satisfy all the target conditions. The page size (k) is set to 10. Each task was completed by 36 MTurk users; we present the average results.

Figures 8(a) and 8(b) show the average time taken and average number of actions executed respectively, by users to find the best target object. As seen in Figure 8(b), if we have more target conditions (e.g. Q_{19}, Q_{26}), using *DIV* (diversity-only), the chances of getting a desired target condition on a given page increases. This increases the probability of *REFINE* action and, therefore, *DIV* performs better (Figure 8(b)) than the other two baselines *REL, MMR,* and slightly worse that our algorithm *ADA,* which prefers diversity over relevance during the initial steps. If we decrease the number of target conditions, the performance of *DIV* degrades, especially if multiple results satisfy all target conditions (as seen for the other queries), since the user needs to find the best and not any object. For Q_{17} and Q_{24}, *MMR* ($\lambda = 0.5$) and *ADA* perform similarly, which intuitively shows that 0.5 happens to be the ideal balance between relevance and diversity for these two queries. This is clearly not that case for other queries such as Q_{23}, where *MMR* takes longer time even compared to the relevance-based *REL*.

As shown by the average values in Figure 8(a), *ADA* reduces the navigation time significantly compared to all other algorithms. On average, *ADA* is faster by a factor of 2.04 (p-value 0.004), 1.97 (p-value 0.001) and 1.66 (p-value 0.0002) over *REL*, *MMR* and *DIV,* respectively. This significant improvement is because of the smaller number of actions incurred by *ADA* compared to the other three algorithms (Figure 8(b)).

Figure 9 shows the actual user navigation time vs. the estimated cost (using our cost model in Section 3) for the 10 different queries using the four different algorithms, where for each query we average over all users. The figure and the trend line show a clear correlation, and specifically a linear relationship, which confirms the validity of the cost model.

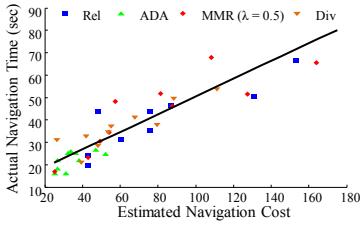

Figure 9. Estimated Cost vs. Actual Time

7. RELATED WORK

Diversification has recently been introduced in search engines and recommendation systems to increase user satisfaction. [6][17] use coverage approaches to cover diverse aspects of search space. [17] expresses the degree of diversification by a setting a parameter which also determines the size of final result set. [6] operates on Web documents and selects diverse documents to cover many different interpretations of the query. [18] provides diversity using the content of the recommendations and the past history of the user. [19] presents a method based on medoids clustering to select a set of diverse and highly-ranked items to recommend to a user. However, none of these approaches considers the problem of how to minimize the total navigation cost incurred by the user to find the target object by considering subsequent navigation steps. In this paper, we have addressed this concern by introducing a navigation cost model (Section 2).

Diversification is being used along with *relevance* in [1][3][5][10]. Most of these approaches (e.g. [5]) consider

diversification as a bi-criteria optimization, which uses a fixed trade-off value for relevance and diversity. In Section 5, we have shown that different search tasks require different ideal trade-off value. Some learning methods [4][16] have been proposed to learn the trade-off value. However, these methods rely on training data provided by the experts which is expensive and difficult to obtain in all cases. [25] proposes a dynamic ranking model for interactive user feedback. The ranking is based on the history of other users. In contrast, our ranking function aims to minimize the expected user effort. Threshold based techniques, a variant of the optimization problem, have been proposed in [11] to solve the diversification problem. These approaches consider a threshold value of relevance and maximize the diversity between results (or vice versa). Setting a threshold value is hard, and depends on the domains. Moreover, [5] shows a comparison between several diversification approaches where *MMR* (the baseline used in this paper) clearly outperforms the threshold based approach in both quality and time. Our approach does not require the threshold value, and can adaptively set the balance between relevance and diversity for different tasks.

Faceted Search has been shown to be effective in reducing the user effort and time required to navigate large result sets of structured databases. For a given query, these approaches compute the best facet conditions and matching results to display to the user [12][21][22]. However, this model is not suitable for our setting of limited screen size, where we cannot display separately faceted conditions and results. Instead, our results serve a dual purpose, since a user can click on results' conditions to refine her navigation.

8. CONCLUSIONS

We described a novel framework for adaptive diversification of query results that dynamically adjusts the relevance and diversity of displayed results with the aim to minimize the total expected user navigation cost to reach the desired target objects. Based on this framework, we prove that the problem is NP-hard and we present an efficient approximate algorithm (*ADA*) that computes the best set results to display, by dynamically balancing relevance and diversity at each query step. We experimentally evaluate the performance of our proposed algorithm and show that it outperforms state-of-the-art algorithms. A Mechanical Turk user study confirms our findings and validates our navigation model. As a future work, we plan to extend our navigation model where the user can update/delete the selected conditions during the navigation process.

9. ACKNOWLEDGMENTS

This project was partially supported by NSF grants IIS-1161997, IIS-1216007, and a Samsung GRO grant.

10. REFERENCES

[1] C. Clarke, M. Kolla, G. Cormack, O. Vechtomova, A. Ashkan, S. Buttcher, I. MacKinnon. Novelty and Diversity in Information Retrieval Evaluation. In *SIGIR 2008*.

[2] H. Tong, J. He, Z. Wen, R. Konuru, and C. Lin. Diversified Ranking on Large Graphs: An Optimization Viewpoint.In *KDD 2011*.

[3] E. Demidova, P. Fankhauser, X. Zhou, and W. Nejdl. DivQ: Diversification for Keyword Search over Structured Databases. In *SIGIR 2010*.

[4] R. Santos, C. Macdonald, and I. Ounis. Selectively Diversifying Web Search Results. In *CIKM 2010*.

[5] M. Vieira, H. Razente, M. Barioni, M. Hadjieleftheriou, D. Srivastava, C. Traina, and V. Tsotras. On Query Result Diversification. In *ICDE 2011*.

[6] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying Search Results. In *WSDM 2009*.

[7] F. Radlinski, P. Bennett, B. Carterette, and T. Joachims. Redundancy, Diversity and Interdependent Document Relevance. In *SIGIR Forum,* 43(2):46-52, *2009*.

[8] S. Chaudhuri, G. Das, V. Hristidis, G. Weikum. Probabilistic Ranking of Database Query Results. In *VLDB 2004*.

[9] K. Raman, P. Shivaswamy,and T. Joachims. Online Learning to Diversity from Implicit Feedback. In *KDD 2012*.

[10] J. Carbonell and J. Goldstein. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *SIGIR 1998*.

[11] L. Qin, J. X. Yu, and L. Chang: Diversifying Top-K Results, In *VLDB 2012*.

[12] A. Kashyap, V. Hristidis, and M. Petropoulos. FACeTOR: Cost-Driven Exploration of Faceted Query Results. In *CIKM 2010*.

[13] M. J. McGill, G. Salton. Introduction to Modern Information Retrieval. McGraw-Hill, 1986.

[14] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. A. Yahia. Efficient Computation of Diverse Query Results. In *ICDE 2008*.

[15] K. Liu, E. Terzi, and T. Grandison. Highlighting Diverse Concepts in Documents. In *SDM 2009*.

[16] Y. Yue, and T. Joachims. Predicting Diverse Subsets Using Structural SVMs. In *ICML 2008*.

[17] M. Drosou, E. Pitoura: DisC Diversity: Result Diversification based on Dissimilarity and Coverage, In *VLDB 2013*.

[18] C. Yu, L. Lakshmanan, and S. A.Yahia. It Takes Variety to Make a World: Diversification in Recommender Systems. In *EDBT 2009*.

[19] R. Boim, T. Milo, S. Novgorodov. Diversification and Refinement in Collaborative Filtering Recommender. In *CIKM 2011*.

[20] C.-N. Ziegler, S. McNee, J. Konstan, G. Lausen: Improving Recommendation Lists through Topic Diversification, in *WWW 2005*.

[21] S. B. Roy, H. Wang, G. Das, U. Nambiar, and M. Mohania. Minimum-Effort Driven Dynamic Faceted Search in Structured Databases. In *CIKM 2008*.

[22] K. Chakrabarti, S. Chaudhuri, and S. Hwang. Automatic Categorization of Query Results. In *SIGMOD 2004*.

[23] M. Hasan, A. Mueen, V. Tsotras and E. Keogh. Diversifying Query Results on Semi-Structured Data. In *CIKM 2012*.

[24] M. Hasan, A. Mueen and V. Tsotras. Distributed Diversification of Large Datasets. In *IC2E 2014*.

[25] C. Brandt, T. Joachims, Y. Yue and J. Bank. Dynamic Ranked Retrieval. In *WSDM 2011*.

[26] Amazon Mechanical Turk. https://www.mturk.com

[27] eMarketer Report http://www.emarketer.com/Article/Smartphones-Tablets-Drive-Faster-Growth-Ecommerce-Sales/1009835, 2013

[28] Extended Version of the paper appears at: http://www.cs.ucr.edu/~hasanm/adadiv.pdf