Toward Self-Correcting Search Engines: Using Underperforming Queries to Improve Search

Ahmed Hassan
Microsoft Research
Redmond, WA 98052 USA
hassanam@microsoft.com

Ryen W. White
Microsoft Research
Redmond, WA 98052 USA
ryenw@microsoft.com

Yi-Min Wang
Microsoft Research
Redmond, WA 98052 USA
ymwang@microsoft.com

ABSTRACT

Search engines receive queries with a broad range of different search intents. However, they do not perform equally well for all queries. Understanding where search engines perform poorly is critical for improving their performance. In this paper, we present a method for automatically identifying poorly-performing query groups where a search engine may not meet searcher needs. This allows us to create coherent query clusters that help system designers generate actionable insights about necessary changes and helps learning-to-rank algorithms better learn relevance signals via specialized rankers. The result is a framework capable of estimating dissatisfaction from Web search logs and learning to improve performance for dissatisfied queries. Through experimentation, we show that our method yields good quality groups that align with established retrieval performance metrics. We also show that we can significantly improve retrieval effectiveness via specialized rankers, and that coherent grouping of underperforming queries generated by our method is important in improving each group.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *selection process, search process.*

Keywords

Dissatisfied query groups; Search satisfaction; Specialized rankers.

1. INTRODUCTION

Search engines are emerging as the primary resource through which people find information. As a result, they are expected to handle a broad range of different types of requests. However, search engines may only apply a single ranking function over all queries they receive [35]. Inevitably in such circumstances, there are queries that search engines do not handle well, leading to user dissatisfaction (DSAT). Automatically identifying groups of underperforming DSAT queries could help search designers decide where to focus resources to improve performance.

There has been work on studying searcher success and frustration as well as measuring query performance through human labels or via automatic methods [13][14][20]. Those methods have focused on identifying individual instances of user dissatisfaction from log data. However, from the search engine's perspective it is not cost effective to simply identify particular queries where the engine performs poorly. Commercial search engines are trained over very large sets of queries. When a poorly performing query is identified, this signal can be difficult to use for improving ranking quality. First, it is only a single example, among millions of others, so even if it is added to the training data, it is unlikely that it will have any effect on the training model. Moreover, machine learned models are difficult to debug or interpret and hence we will most likely be unable to determine why a particular query fails. Finally, human intervention, either for the generation of new interface, for ranking features, or for manually labeling more examples of queries, is costly and it is not prudent to make these investments on a per-query basis. Therefore, we must be able to combine the queries into coherent groups to take action and improve them.

In this paper we present an approach for identifying groups of queries where search engines perform poorly and develop methods to automatically improve retrieval quality for groups of failing queries. Dissatisfied query groups are identified automatically from search engine logs. The resulting groups have a strong correlation with two established measures of search performance: search-result clickthrough rate (CTR) and normalized discounted cumulative gain (NDCG) [26]. The query groups that our method identifies therefore comprise queries that are likely to result in searcher dissatisfaction. The queries in these groups share common attributes and are coherent in a way that makes them susceptible to ranking improvements. As such, we also trained a specialized ranker that is optimized to a particular dissatisfaction group that yielded significant relevance gains over our baselines.

This paper makes the following research contributions:

- Proposes the automatic identification of underperforming (dissatisfaction) query groups as a means for search engines to better understand and improve their performance.
- Introduces a novel method for automatically identifying such dissatisfaction groups given interaction log data.
- Validates that the groups that our method identifies are indeed correlated with dissatisfaction estimates made via established measures of search performance.
- Develops specialized rankers that leverage the coherence of the queries in the dissatisfaction groups to improve the search engine's retrieval quality.
- Demonstrates through experimentation that: (i) our learned specialized ranker outperforms a general ranker trained on a diverse set of queries, and (ii) the coherence of the groups identified by our method leads to larger relevance gains over a specialized ranker trained only on underperforming queries.

The remainder of this paper is structured as follows. Section 2 describes relevant related work in the areas of search success and query performance. Section 3 defines our problem and Section 4 describes how we extract instances of dissatisfaction from log data. We describe our method for automatically grouping dissatisfaction instances in Section 5. In Section 6 we describe the training of a specialized ranker to target a typical DSAT group identified using our method. Experiments and associated results are presented in Section 7. We discuss the findings and their implications in Section 8 and conclude in Section 9.

2. RELATED WORK

There are two areas of work related to the research presented in this paper: (i) search satisfaction, success, and frustration, and (ii) query performance. We cover each of the two areas in turn.

2.1 Satisfaction, Success, and Frustration

Extensive literature exists on deriving indicators of task success or failure from online user behavior. In search specifically, several fruitful approaches have been tried. One approach is to correlate behavior with either self-reported success [14] or labels of success provided by expert judges [13][20]. Early investigations correlated self-reported measures of search satisfaction with implicit signals, such as search-result clicks and dwell time for clicks [14].

Fox et al. [14] used an instrumented browser to determine whether there was an association between explicit ratings of satisfaction and implicit measures of searcher interest and identified the measures that were most strongly associated with user satisfaction. They found that there was a link between user activity and satisfaction ratings, and that clickthrough, dwell time, and session termination activity combined to make good predictors of satisfaction for Web pages. Fox et al. found that short dwell times and clicking many (four or more) search results for a query were both indicators of dissatisfaction. Behavioral patterns were also used to predict user satisfaction for search sessions. Feild et al. [13] developed methods to predict user frustration. They assigned users difficult information seeking tasks and monitored their degree of frustration via query logs and physical sensors. One behavior that can be associated with dissatisfaction is search engine switching – the voluntary transition between different engines. Guo et al. [16] characterized the reason that searchers switch between search engines. Using a browser plugin, they captured switching rationales in-situ. They showed that one of the primary reasons that searchers changed engine was dissatisfaction with the results provided by the pre-switch engine. Using the labeled switching events gathered from the plugin, they also showed that there was sufficient consistency in searchers' interaction behaviors before and after the switching event to accurately predict switching rationales. This predictor can be used to dynamically adapt the search experience and derive more accurate competitive metrics. Indeed, Feild et al. showed that features capable of accurately predicting switching events were also highly predictive of searcher frustration.

Huffman and Hochster [25] found a relatively strong correlation with session satisfaction using a linear model encompassing the relevance of the first three results returned for the first query in a search task, whether the information need was navigational, and the number of events in the session. Hassan et al. [20] developed models of user behavior to accurately estimate search success on a session level, independent of the relevance of documents retrieved by the search engine. Ageev et al. [1] propose a formalization of different types of success for informational search, and presented a scalable game-like infrastructure for crowdsourcing search behavior studies, specifically targeted towards capturing and evaluating successful search strategies on informational tasks with known intent. They show that their model can predict search success effectively on their data and on a separate set of log data comprising search engine sessions. Going beyond individual search sessions to study the effects of satisfaction over time, Hu et al. [23] performed a longitudinal study of the relationship between search satisfaction and search engine re-use. They showed that satisfaction with a search engine can contribute to users' propensity to use and re-use its service over time.

2.2 Query Performance

Search engine performance for a particular query is typically measured using relevance metrics such as NDCG [26]. NDCG can be calculated based on manual judgments on the relevance of documents in the result list, or estimated using models derived from user click behavior (e.g., [2]). Session discounted cumulative gain (SDCG) [27] applies a discount to relevant results found in response to queries later in the session. This considers that multiple queries can be part of a search goal, but still requires manual relevance judgments, which can be costly to obtain and do not scale to unseen queries. Al-Maskari et al. [4] found a reasonable correlation between many information retrieval (IR) metrics and user satisfaction with result rankings (although surprisingly not NDCG, given the small numbers of judgments available per query). They advocated for a combination of different measures in evaluating the effectiveness of IR systems.

Research on predicting query performance has been conducted to understand differences in the quality of search results provided by search systems for different queries. Measures such as Jensen-Shannon divergence [8], query clarity [10], and weighted information gain 0 have been developed to predict the retrieval performance on a query (as measured by average precision, for example). He and Ounis [23], and Hauff et al. [22] also developed automatic methods for predicting query difficulty (i.e., how good the search results are for a query). Leskovec et al. [29] used graphical properties of the link structure of the result set to predict the quality of the result set and the likelihood of query reformulation. Teevan et al. [31] developed methods to predict which queries could most benefit from personalization. Research has also been conducted on predicting query performance using searcher interactions. Carterette and Jones [9] used clickthrough behavior to evaluate the quality of search advertising results, but they did not study other interaction features, and their focus was on search advertising not general Web search. Guo et al. [14] used interaction features, including switching features, to predict query performance. However, that was on a per-query basis rather than on the basis of query groups that lead to searcher dissatisfaction.

The research presented in this paper extends this previous work in a number of ways: (i) rather than predicting the performance of individual queries, we focus on identifying underperforming query *groups* that share attributes such as length or domain; (ii) we validate that the groups represent likely searcher dissatisfaction via established performance measures such as clickthrough rate and NDCG, and; (iii) we train a specialized ranker for groups identified and show that (a) we can obtain relevance gains by targeting specific DSAT groups, and (b) training a specialized ranker tailored to the coherent DSAT groups identified by our method is better than simply targeting the broad set of all underperforming queries (e.g., only those queries with low CTR).

3. PROBLEM DEFINITION

We start by defining some terms used throughout the paper:

DEFINITION. A *query group* is a set of queries sharing a set of common factors and can be characterized using binary attributes. For example "Queries about movies with 5 or more words".

DEFINITION. A *DSAT query group* is a set of queries for which the search engine performs poorly in comparison to its average performance over all queries.

DEFINITION. A *specialized ranker* is a ranking model that is built to handle a specific query group. The specialized ranker has a local effect because it only provides the search results for queries belonging to a specific query group.

Given a stream of queries submitted by search engine users, we seek to: (i) automatically identify poorly-performing groups of queries suggestive of searcher dissatisfaction (DSAT), and (ii) train a specialized local ranker capable of addressing the DSAT through relevance gains. To identify the underperforming query groups we construct attribute sets that cover DSAT instances and correlate highly with dissatisfaction.

Commercial search engines rely on features of log data. They also need humans to define these features and to generate relevance labels from which the ranking algorithms can learn [2]. In theory, a search engine that adopts a framework capable of doing both (i) and (ii) above could improve with little need for human involvement in generating additional features which could result in a self-correcting search engine. The development of such a framework motivates our research. We now describe its implementation in our context.

The input to our method is the search log interaction data gathered from consenting users of a toolbar deployed by a commercial search engine. Search logs are usually organized in the form of search sessions. A *search session* is a sequence of user activities that begin with a query, includes subsequent queries and URL visits, and ends with a period of inactivity. A session ends if the user was idle for more than 30 minutes on a page. The 30-minute cutoff to determine session boundaries has been commonly used in previous work, e.g., [12]. In our case, search sessions can span multiple search engines since we are interested in transitions between engines as evidence of searcher dissatisfaction. More discussion on this issue is provided in the next section of the paper.

We use the data from search sessions to automatically identify queries that resulted in searcher dissatisfaction. This step is the first step in our process. This module uses log data (from the search engine or in our case a toolbar from the search provider) to generate a large number of individual dissatisfaction instances. For each instance, we collect information about the query, time and market where it was submitted, and the search engine results page (SERP). Individual dissatisfaction instances were used to identify poorly performing groups of queries.

Specialized rankers specifically targeting the identified groups are then trained to improve the search experience. One specialized ranker is trained for each dissatisfaction group. The specialized ranker can use the same features used by the original ranker, but trained using a group-specific dataset. Alternatively, additional investment may be involved to further improve these groups. For example, once the groups have been identified, additional features may need to be generated to help the ranker learn how to perform better for this query group. This activity is labor intensive and time consuming, meaning that an automated alternative such as the method proposed in this paper can be extremely valuable.

Finally the specialized ranker is combined with the general ranker and the resultant ranker can be used to replace the original. The specialized ranker has a local effect because it will be used for queries that belong to the corresponding group only. The general ranker will be used for the remainder of the query traffic. If gains are achieved through the specialized ranker and all other queries are treated in the same way as before, then overall relevance gains will be observed. *The primary challenge is therefore in identifying dissatisfaction groups because enumerating all possible ways of combining attributes to form groups is prohibitively expensive. Addressing this challenge is our focus in this paper.*

We begin by describing how we identify the individual DSAT instances that are used to construct the DSAT query groups.

4. MINING DSAT INSTANCES

We mine instances of searcher dissatisfaction based on search engine switching events. Search engine switching is the process describing a user's voluntary transition from one Web search engine to another. A search engine switching event is a pair of consecutive queries that are issued on different search engines within a single session. Note that in identifying pre-switch queries, if the user issued a navigational query for a target search engine (e.g., search for "yahoo" on Google, or "google" on Bing), this query is regarded as part of the switching action and the preceding query in the pre-switch engine is used as the "pre-switch" query.

Search logs contain a large and varied set of search behaviors that could be associated with searcher dissatisfaction (e.g., clickthrough rate, abandonment, short dwell time or quick back clicks). We use search engine switching to identify user dissatisfaction instances for two main reasons: (i) search engine switching is a rare but important event that has been shown to correlate with dissatisfaction in prior research [16][32], and we have a reliable classifier that can distinguish switches caused by dissatisfaction from others; (ii) other query performance measures such as clickthrough data or editorial judgments could be used, but we abstained from using them in this phase in order to be able to use them later to validate our results. After the validity of our dissatisfied query groups is established through our study, these query performance measures could be used to generate more dissatisfaction instances.

Users switch from one search engine to another for various reasons. One of the most common reasons is dissatisfaction with the results they received from the pre-switch engine [33], which accounts for around 60% of engine switching events. Guo et al. [16] proposed a method for estimating the cause behind an observed switch given recent search behavior such as queries and SERP clicks on the source and destination search engines. They gathered ground truth data through the deployment of a plugin to around 200 Web searchers. The plugin captured switching rationales in-situ when a switch occurred and also logged search behavior before and after the switch. The cause of the switch could be one of: (i) dissatisfaction (the searcher was unhappy with the pre-switch engine), (ii) coverage (the searcher wanted to check the information they had found on the other search engine), (iii) preferences (they usually used the target engine or the target engine was better for the current task type), (iv) unintentional (browser defaults or homepage settings), and (v) other. They found that they could accurately predict when the reason for an engine switch was dissatisfaction-related.

Inspired by the work of Guo and colleagues, we built a classifier to distinguish dissatisfaction-related engine switches. The dissatisfaction engine switching classifier learns to predict the switch cause from training data of switching instances from [16]. The dataset contained 562 labeled switches from 107 different users.

We merged all switching causes into two classes: dissatisfaction and everything else, and trained a logistic regression classifier to identify dissatisfaction instances.

DEFINITION. A *dissatisfaction instance* in our study is a switch between search engines where the pre- and post-switch queries are the same and has been labeled by our classifier as DSAT-related. We assume that a dissatisfaction instance represents dissatisfaction with the pre-switch engine.

We represent search behavior by adapting and extending a subset of the features presented in previous studies that we believed were particularly likely to be associated with dissatisfaction [16]. We

Table 2. Features to identify dissatisfied engine switches.

Feature	Description
Query features	
$char\_len$	Num. characters in query
$word\_len$	Num. words in query
$time\_diff$	Time in seconds between pre-switch and post switch queries
Pre- and Post-switch features	
$num\_queries$	Num. queries in session
$uniq\_queries$	Num. unique queries in session
num_reform	Num. query reformulations
num_clicks	Num. clicks
num_sat_clicks	Num. clicks with dwell time > 30s
num_quick_backs	Num. clicks with dwell time < 15s
num_term_clicks	Num. clicks on URLs containing a query term in their title
$trail\_len$	Length of the trail from search engine result page, defined as the number of clicks made after leaving SERP
a_{ij}_count	Num. transitions between every action pair $a_i \rightarrow a_j$ for every $a_i \in A$ where A = {Query, SERPClick, AdClick, Answer, etc.}
a_{ij}_time	Avg. dwell time for every action pair $a_i \rightarrow a_j$

also added features describing the action transition of users (e.g., query-click, query-query, etc.), and the time difference between every pair of actions. It has been shown in previous work that action transitions are a very good descriptor of user behavior when modeling satisfaction [20][21]. An action could be a query submission, a click on a SERP, or a click on any other SERP feature. For example, submitting a query followed by reformulating and resubmitting a related query is a sign of dissatisfaction. The features used in our classifier are summarized in Table 1. We only considered switching instances that had the same query before and after the engine switch since these have been shown to be more strongly associated with dissatisfaction in previous work [16].

Later in our experiments, we will use only query-related features and post-switch features to identify switches resulting from DSAT. Excluding pre-switch features allows us to validate the results of our DSAT group identification using measures such as SERP click-through rate without introducing any bias.

We evaluated the performance using the F-score, with β set to 0.5. This gives twice as much weight to precision than to recall. We are interested in a highly precise set of dissatisfaction cases that we can reliably use as input for the following steps. There are a large number of switching instances in the logs. We need to precisely identify some of them, rather than finding many of them with lower precision. We evaluated the performance using 10-fold cross validation which resulted in an $F_{0.5}$ score of 81.2 when we use query and post-switch features. This exceeds the 79.0 reported by Guo et al. [16], attributable to the extra features we added.

5. IDENTIFYING DSAT GROUPS

Many previous studies have addressed the problem of measuring query performance [10][26][29]. Individual instances where users are dissatisfied with search result quality can be identified using a number of methods. One such method was presented in the previous section. Given a set of DSAT instances, the more challenging problem is how to take actions to improve search relevance based on these observations. This is challenging because it may be difficult to tweak the search engine ranking algorithm to handle a dissatisfaction case in an isolated context. To provide richer context to dissatisfaction cases, we seek to find frequently-occurring patterns that define dissatisfaction *groups*.

In the remainder of this section we describe the methods that we use to identify dissatisfaction groups. We begin with how we represent the individual DSAT instances, then describe the frequent pattern mining approach that we used to identify frequent attribute sets, and conclude by describing the method used to create the dissatisfaction groups from these attribute sets.

5.1.1 Representation

We describe every dissatisfaction instance with a vector of binary attributes. Each attribute describes a specific characteristic of the dissatisfaction instance. Let $A = \{a_1, a_2, \dots, a_n\}$ be a set of n binary attributes. Let $D = \{d_1, d_2, \dots, d_n\}$ be a set of dissatisfaction instances. Every DSAT instance in D has a subset of attributes in A.

We use several types of attributes to describe each dissatisfaction instance. We summarize them below:

Query Attributes: This is a set of attributes that describe the query itself. Several categories are used to describe the intent behind the query. It is impractical to use the text in Web pages to draw conclusions about the topicality of the query. Conversely, we could look at URLs or domains but that would be very limited due to data sparseness. Instead, we used the Open Directory Project (ODP), also referred to as dmoz.org. ODP is an open Web directory maintained by a community of volunteer editors. It uses a hierarchical scheme for organizing URLs into categories and subcategories. We use the top two ODP categories and assign labels to URLs automatically using an approach similar to [34]. URLs that exist in the directory were classified according to the corresponding categories. Missing URLs were incrementally pruned one path level at a time until a match was found or a miss declared. A query is assigned the plurality label of the labels of its top 10 results. In addition to these categories, we also include the query length, language, query phrase type (noun phrase, verb phrase or a question) as determined by the Stanford parser [28].

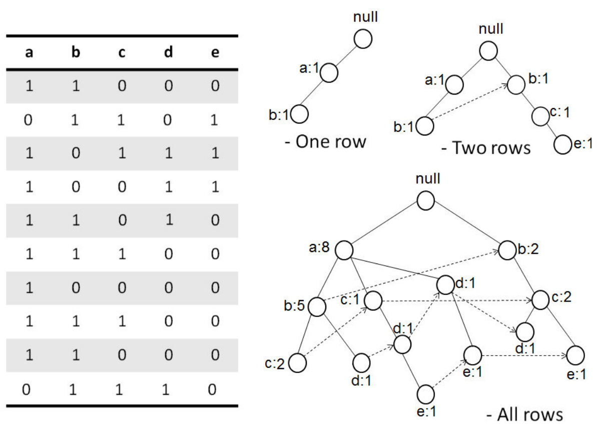
SERP Attributes: We also added a set of attributes to describe the SERP that was displayed to the user as a response to submitting the query. Examples of these attributes include whether a direct answer was displayed for that query, and if so, then what type of answer (e.g. Weather, Stocks, etc.), whether a query suggestion was shown, and whether a spelling correction was shown.

Impression Attributes: An impression is a single instance of a particular query. We used another set of attributes to describe each dissatisfaction impression. We use the market from which the query was issued (e.g., US, UK, etc.), the vertical used to issue the query (e.g., Web, News, Images, etc.) and the search engine used to issue the query (Bing, Google or Yahoo). This allows us to identify DSAT groups for specific markets, verticals, and engines as well as DSAT groups that span multiple markets, verticals, or engines. We also added some temporal attributes such as day of the week, time of day (morning, afternoon, evening, night), and month of the year.

This yields a set A with 140 attributes. Every dissatisfaction instance d is represented with a subset of attributes in A.

5.1.2 Frequent Patterns

Identifying frequent patterns in transactional or relational data sets is a well-studied problem in the data mining literature [18]. The problem was motivated by the massive amounts of data continuously collected and stored by many businesses. The discovery of



a	b	c	d	e
1	1	0	0	0
0	1	1	0	1
1	0	1	1	1
1	0	0	1	1
1	1	0	1	0
1	1	1	0	0
1	0	0	0	0
1	1	1	0	0
1	1	0	0	0
0	1	1	1	0

Figure 1. An example illustrating how the FP-tree is constructed in the FP-Growth algorithm.

frequent patterns in this huge amount of data was found useful for many decision making processes. A typical example of this scenario is the market basket analysis problem. The objective of this analysis is to find associations between the different items that customers purchase. This information is valuable for designing marketing plans and product catalogs.

The problem of mining dissatisfaction groups can be approached from a similar perspective. To find dissatisfaction groups, we look for frequent patterns of attributes that correlate with user dissatisfaction. The representation we described in the previous subsection aligns well with this approach. In the proposed representation, every dissatisfaction instance is represented by a row, and every attribute is presented by a column. All values are binary; indicating whether the corresponding attribute is present in a particular row or not. Multi-valued attributes are replaced with multiple variables corresponding to the different values. An attribute in our case is analogous to an item in the market basket analysis case. Note that we use the terms "attribute" and "item" interchangeably throughout the remainder of the paper.

There are several algorithms for discovering frequent patterns from relational databases. For example, Apriori [5] is a seminal algorithm for finding frequent itemsets. It employs a level-wise search process where frequent itemsets of size l are used to discover frequent itemsets of size $l + 1$. It starts by finding the counts of single items, then itemsets with size 2 and so on. Finding each set of itemsets with a certain size requires a full scan of the dataset. To reduce the search space, it discards all itemsets that are supersets of previously found infrequent itemsets.

The main disadvantage of Apriori is that it employs a generate-and-test method which may result in the generation of a huge number of candidate sets. Additionally, it may need to repeatedly scan the database for every itemset size. The FP-Growth algorithm [19] overcomes these problems via a divide-and-conquer strategy. FP-Growth only needs to scan the dataset twice, and unlike Apriori, it does not involve any candidate itemset generation. We use the Weka [17] implementation of FP-Growth [19]. There are also variants of FP-Growth that were designed to be run in parallel on a distributed cluster of machines, allowing large volumes of log data to be processed using this algorithm [11][30].

The FP-Growth algorithm consists of two main steps. In the first step, a compact data structure, the FP-tree, is built to represent the data. In the second step, frequent patterns are extracted from the FP-tree. To build the FP-tree, the data are scanned twice. In the first scan, it counts the frequencies of every attribute and discards infrequent attributes. The attributes are then sorted in decreasing order based on their frequencies. This fixed order is used so paths can overlap when records share items. So, paths overlap when they share the same prefix.

Table 2. Examples of frequent attribute sets.

Frequent Attribute Sets
{English Query, Local, Navigational, NumWords >10}
{English Query, Locations, Questions, NumWords >10}
{English Query, US Market}
{NumWords >10}

Table 3. Dataset contains satisfaction and dissatisfaction.

Attribute_1	Attribute_2	…	Attribute_n	(D)SAT
True	False	…	True	DSAT
True	False	…	False	DSAT
False	True	…	True	DSAT
…….	…….	…	…….	…….
False		…	True	SAT
True	True	…	False	SAT
True	False	…	True	SAT
…….	…….	…	…….	…….

In the second scan, the FP-tree is constructed. The FP-tree is a compact way to retain information about itemset association [19]. Nodes in the tree represent attributes and each node has a counter. The algorithm reads one row at a time and maps it to a path. Paths overlap when rows share attributes and node counters are incremented. Special pointers are used to link nodes representing the same attribute. An example illustrating the FP-tree construction is shown in Figure 1. The algorithm starts by creating the root of the tree and labeling it "null". For each row in the dataset, the items in that row are sorted according to their frequency and a branch is created to represent the row. Every node in the branch corresponds to an item and is labeled with the item identifier and its count; which is initialized to 1 at creation. This process is repeated for every row. A pointer is created to connect any two nodes with the same identifier and the node count is updated accordingly. Figure 1 shows the tree after reading the one row, two rows and all rows. After reading the first row, where only *a* and *b* are set to 1, two nodes are added to the tree corresponding to *a* and *b* and the counter is set to *1* in both cases. The second row has *b, c,* and *d* set to *1*. The second row does not share the same prefix with the first one, hence a new path is created and the counters of all nodes are set to *1*. When the algorithm reads the third row, it finds out that the prefix {a} overlaps with the prefix for the first row. Hence, the counter of node *a* is increased and new nodes are created to represent *c*, *d*, and *e*. This process continues until all rows are scanned. Pointers are maintained between nodes representing the same item (dotted lines in the figure). As explained earlier, the algorithm needs to scan the dataset only twice. In the first scan, the number of occurrences of every attribute is computed and infrequent attributes are discarded. The FP-tree is created in the second scan.

After constructing the FP-tree, we proceed to the second step. To extract frequent patterns, FP-Growth applies a bottom-up algorithm using a divide-and-conquer strategy starting at the leaves and moving toward the root, building frequent patterns as it ascends the tree. For example, it starts with frequent attribute sets ending in *e* and then ending in *de* and so on so forth.

We apply the FP-Growth algorithm to the dataset described in the previous section. Table 2 presents a few examples of some of the identified frequent attribute sets. Some sets consist of a single attribute, while others have many attributes. In the next subsection, we describe how dissatisfaction groups can be identified using these frequent attribute sets.

5.1.3 From Frequent Patterns to DSAT Groups

Frequent attribute sets discovered from a set of dissatisfaction instances do not necessarily define a dissatisfaction group. It could be the case that a set of attributes defines a large query group, and even though the percentage of dissatisfaction cases resulting from this group is very small, it could still be overrepresented in the dissatisfaction dataset. For example, FP-Growth reported that the group $\{English\ Query, US\ Market\}$ is frequent. This occurred because most of the queries in the dataset are English queries coming from the US market, not because this is a dissatisfaction group.

To address this problem, we built a new data set, the *satisfaction dataset*, and define a measure to identify groups correlated with dissatisfaction. We build this dataset by randomly sampling a number of queries from L where: (i) the query received many clicks and the last one had dwell time greater than 30 seconds, or (ii) the query received a single click with dwell time greater than 30 seconds. Dwell time greater than 30 seconds has been widely used to denote satisfaction as in previous work [14].

Table 3 shows an example illustrating the appearance of the combined dataset. The dataset contains both satisfaction (SAT) and DSAT instances. Every instance is represented by a row in the dataset. The same set of attributes is used to describe both types of instances. An additional attribute is added to show whether the instance is associated with satisfaction or dissatisfaction.

We apply the FP-Growth algorithm to the data to generate sets of attributes that co-occur frequently with one another. Unlike the market basket analysis, we are not concerned with the correlation between all kinds of attributes. We are mainly interested in attribute sets that correlate highly with dissatisfaction. As a result, we restrict the generated sets to those that include the attribute "DSAT" and ignore all other patterns.

To distinguish attribute sets that are highly correlated with dissatisfaction and attribute sets that are highly represented in both satisfaction and dissatisfaction instances, we define "*DSAT correlation*". *DSAT correlation* is a simple measure that estimates the correlation/dependence between an attribute set and dissatisfaction. It is defined as follows for any attribute set $A = \{a, \dots, a_n\}$:

$$DSAT\ correlation = \frac{P(A, DSAT)}{P(A)P(DSAT)}$$

where $P(X)$ is the probability of observing the set of attributes $P(X)$. $P(A, DSAT)$ is estimated by dividing the number of instances that belong to both the group defined by A and the dissatisfaction set by the total number of SAT and DSAT instances. $P(A)$ is estimated by dividing the number of instances that belong to the group defined by A by the total number instances. Finally $P(DSAT)$ is the proportion of dissatisfaction instances in the dataset. The number of occurrences of A and $A \cap DSAT$ are computed using the FP-Growth algorithm as explained earlier.

If the resulting value of the *DSAT correlation* is less than 1, then the occurrence of A is negatively correlated with dissatisfaction. If the resulting value is greater than 1, then A is positively correlated with dissatisfaction. If the value is exactly 1, then A and dissatisfaction are independent. This value is sometimes referred to as the *lift* in the data mining literature [18]. In our case, it implies that the occurrence of the set of attributes in A "lifts" the occurrence of dissatisfaction.

Now let us revisit the group that was defined by the attribute set $\{English\ Query, US\ Market\}$. Even though this set frequently co-occurs with dissatisfaction, its DSAT correlation is 0.98 showing that it is nearly independent of SAT or DSAT. This allows us to identify more interesting groups and gives us a metric to evaluate how every group correlates with dissatisfaction.

Other measures of correlation could be used as well for the same purpose (e.g., χ^2). Correlation measures are better than confidence (i.e., the number of occurrences of $A \cap DSAT$ divided by the number of occurrences of A). Confidence could be misleading if the dataset is unbalanced with respect to the proportion of satisfaction and dissatisfaction instances. This imbalance is typical in any random sample of query impressions because most users achieve their search goals and dissatisfaction is usually the exception. For example, Ageev et al. [1] performed a study where users were asked to find answers to specific questions using Web search. In 87% of the tasks, users reported success, and in 75% of these cases, the answer was correct. Hassan et al. [21] collected firsthand success labels from users using several search engines and reported a 80% satisfaction rate. Notice that the DSAT correlation is also insensitive to the distribution of SAT vs. DSAT cases in the dataset. Hence, it will find attribute sets statistically correlated with dissatisfaction regardless of whether the sample follows the original underlying distribution of satisfaction or not.

6. A SPECIALIZED RANKER

A specialized ranker can be trained for each dissatisfaction group. DSAT groups are easier to learn than individual DSAT instances because they provide more context for the learning algorithm than single instances. Additionally, and importantly, the coherence of the DSAT groups we identify makes them more susceptible to ranking improvements than a potentially broad and heterogeneous set of underperforming queries. We show later in this section that the coherence is important for training specialized rankers.

Attributes of the group can be used to determine for each query whether to apply the method. In practice, the specialized ranker is used for all queries where the group's membership criteria applies and the general ranker is used for all other traffic. If we observe gains through applying the specialized ranker and do not change the general ranker used for all other queries, we will see the overall search effectiveness of the search engine improve. In the remainder of this section we describe how we train a specialized ranker for one such DSAT group identified through the analysis in the previous section and show that we can obtain significant relevance gains over strong baselines by targeting that group.

In a learning-to-rank framework, each query document pair is represented by a feature vector. The performance of any ranking function depends on feature selection and training data. We fix the set of features used for both the general and the specialized rankers. Our feature set contains several hundred features including query, document, and query-document features. It includes many content-based features, click-based features, static rank-based features, and many others. Note that the features that were used to characterize the DSAT groups (i.e., the attributes used by the DSAT group identification step) were also available to the general ranker, giving it an opportunity to learn to use DSAT information on its own and potentially removing the need to identify DSAT groups. The general ranker therefore represents a strong baseline against which to compare our specialized ranking approach.

We employ a re-ranking framework to create a specialized ranker optimized to perform well on the DSAT group. We use a cascade approach where the output of the G*eneral Ranker* is used as a feature for training a Specialized Ranker. The specialized ranker is trained on the DSAT group data only using the same features as the general ranker. The output of the general ranker is added to the training and testing data of the specialized ranker as an additional feature. This allows the specialized ranker to benefit from the signals in the general data, and at the mean time optimize the performance over the specialized data.

Notice that we cannot simply combine the general dataset with the specialized dataset and train a single ranker. By doing this, we are altering the true underlying distribution of queries and the general traffic will be under-represented compared to the natural query distribution. Even though, this ranker may have a better performance on the DSAT group, it will alter the performance of the rest of the traffic. This is an undesirable effect because our objective is to improve the performance on the DSAT group without affecting the general traffic.

By changing the training data for the specialized ranker to only use in-group queries, we show that identifying an underperforming group is a fundamental step toward improving the performance of search engines on queries belonging to this group. We will also show that we can achieve relevance gains (perhaps attributable to group coherence) by simply targeting that group even when the feature set is held constant. Note that the training and testing data does not overlap with the data used to learn the underperforming query groups.

7. EXPERIMENTS AND RESULTS

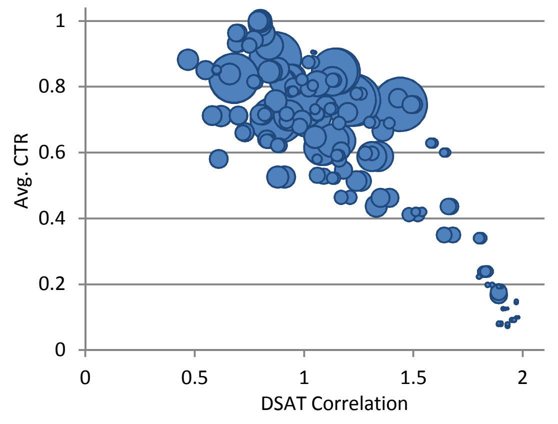
7.1 Validating DSAT Groups

7.1.1 Data and Experimental Setup

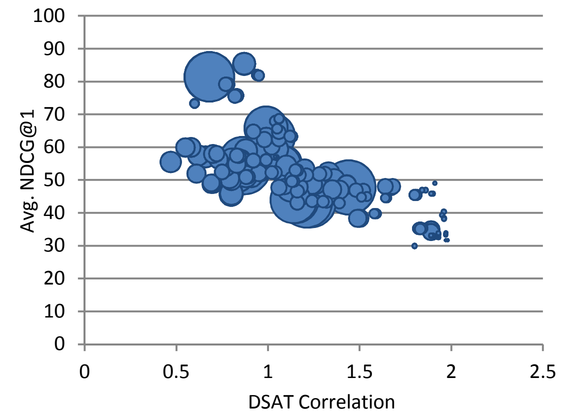
In this subsection, we describe the data we used to identify individual DSAT instances and DSAT groups. We obtained millions of records of interaction logs (from July to September 2011) for hundreds of thousands of consenting users through a widely-distributed Web browser toolbar. Log entries include a unique identifier for the user, a timestamp for each page view, and the URL of the Web page visited. Intranet and secure (https) URL and any personally identifiable information were removed from the logs prior to analysis. From these logs, we identified switching instances where the same query has been issued on different search engines. This set of logs is referred to as L.

We applied the DSAT classifier described earlier to a random set of switches observed in L, setting recall low to boost precision. The classifier predicted that approximately 100,000 of the instances were caused by dissatisfaction with the pre-switch engine results. We randomly sampled a set of another 100,000 satisfaction instances as described in Section 5 to construct the satisfaction dataset and applied the method in Section 5 to generate DSAT groups. We generated around 200 groups; 50% of them were either negatively correlated or uncorrelated with DSAT and were discarded.

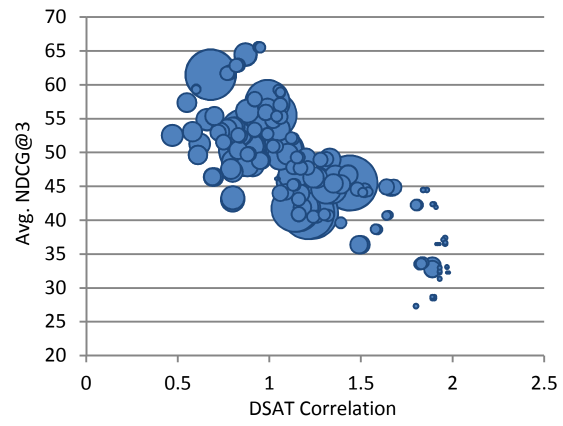
The average number of attributes describing every group was 3.4. The average fraction of dissatisfaction instances in a group was 5.5% of the total number of DSAT instances. We divided the



(a)



(b)



(c)

Figure 2. (a) Relation between average clickthrough rate and DSAT correlation for every DSAT group. (b) Relation between average NDCG@1 and DSAT correlation for every DSAT group. (c) Relation between average NDCG@3 and DSAT correlation for every DSAT group. Note that in all figures the size of the circle denotes the number of DSATs that belong to every group.

Table 4. Correlation between *DSAT correlation* and different query performance measures averaged over queries belonging to the corresponding group.

	Avg. CTR	Avg. NDCG@1	Avg. NDCG@3
DSAT Correlation	–0.87	–0.72	–0.80

Table 5. NDCG improvements for a DSAT query group using a specialized ranker.

NDCG@ 1 Gain	NDCG@ 2 Gain	NDCG@ 3 Gain	NDCG@ 4 Gain	NDCG@ 5 Gain
1.52%	0.58%	0.86%	0.94%	0.77%

Table 6. NDCG improvements for low CTR group using a ranker trained on low CTR data.

NDCG@ 1 Gain	NDCG@ 2 Gain	NDCG@ 3 Gain	NDCG@ 4 Gain	NDCG@ 5 Gain
0.51%	0.39%	0.12%	0.05%	0.03%

groups into three bins according to DSAT correlation: less than 0.8, 0.8 to 1.2, and greater than 1.2. These correspond to groups with negative correlation, no correlation, and positive correlation to dissatisfaction. We noticed that groups with no correlation to dissatisfaction have lower average number of attributes (3.2), and higher average number of instances (7.6%) compared to the other two groups that had a number of attributes of 3.4 and 3.6 and a percentage of instances of 3.8% and 3.7% for the negative correlation and the positive correlation groups respectively.

Once we had the groups generated, we wanted to establish whether they were in fact related to searcher dissatisfaction. To validate that the groups that we identify do actually represent likely searcher dissatisfaction, we used established measures of search engine performance, namely result clickthrough rate (CTR) and NDCG.

For every identified group, we compute the average clickthrough rate for queries belonging to this group. Average clickthrough rate is computed using a separate set of queries that does not overlap with the datasets described in the previous section. Every group is defined using a set of attributes. To compute the average clickthrough rate, we identify all queries that have all those attributes, compute the clickthrough rate for every query and then compute the average over all such queries. We abstained from using any pre-switch features (i.e., only used query and post-switch features; see Section 4) while generating dissatisfaction switches to avoid any bias in the clickthrough rate experiment.

7.1.2 Results

Figure 3a shows the relation between average clickthrough rate and *DSAT correlation* for every group. We included all frequent groups even those who were found to be uncorrelated or negatively correlated with dissatisfaction. Every group is represented by a bubble. The size of the bubble represents the number of dissatisfaction instances that belong to this group. Groups are not orthogonal and hence the same dissatisfaction instance may belong to more than one group. The size of the attribute set defining every group can range from 1 to arbitrary large numbers. In practice, the largest attribute set had no more than six attributes. Groups with size less than 0.5% of the size of the entire dissatisfaction set are ignored. We have to set a lower limit on the size of a group; otherwise every dissatisfaction instance will result in a group.

We can observe from Figure 3a that groups with high *DSAT correlation* have low clickthrough rate, while groups with low *DSAT correlation* have higher clickthrough rate resulting in a high correlation between the two measures. We also notice that most of the large groups are either negatively correlated or uncorrelated with dissatisfaction. Many of those groups are defined by a single attribute such as long queries or informational queries.

Similarly, we compared *DSAT correlation* to NDCG@1 and NDCG@3 in Figures 3b and 3c, respectively. NDCG at a particular rank position k is defined as:

$$NDCG@k = \frac{DCG@k}{IDCG@k} = \frac{1}{IDCG@k} \sum_{i=1}^{k} \frac{2^{rel(i)}}{\log(1+i)}$$

where $rel(i)$ is the relevance of the i^{th} document in the ranked list. NDCG is computed by normalized DCG using the DCG of the ideal ranking (IDCG). We used a large query set with editorial relevance judgments labeled as part of a separate search engine assessment effort to identify a set of queries that belong to each group. Every query-document pair was judged on a five-point grade scale: *Bad*, *Fair*, *Good*, *Excellent*, or *Perfect*, giving us a value for $rel(i)$ for each result. The dataset was constructed using pooled relevance judgments and query-document pairs were judged by trained human assessors. Comparison against NDCG yields very similar results to comparison against CTR.

To quantify the correlation with established query level metrics, we computed the Pearson correlation coefficient between *DSAT correlation* and: (i) average clickthrough rate, (ii) average NDCG@1, and (iii) average NDCG@3. Correlations were measured using the Pearson's correlation coefficient. The results are shown in Table 4. The table shows a strong negative correlation between *DSAT Correlation* and the three other metrics validating that the groups do seem to represent likely searcher dissatisfaction.

In summary, we defined a process and a metric for identifying query groups associated with dissatisfaction. The process uses search interaction logs and does not involve any human intervention. The identified groups show high correlation with established query-level performance metrics such as CTR or NDCG. One important advantage of these groups is that they are coherent in a way that provides a rich context for isolated queries. This coherence allows us to use these groups to improve Web search performance, as we will demonstrate in the next section.

7.2 Specialized Ranker Performance

7.2.1 Data and Experimental Setup

We used the learning-to-rank algorithm presented in [35] and compared different ranking functions using NDCG (defined earlier). In this subsection we present the results of two comparisons: (i) a specialized ranker trained on one of our groups versus general ranker trained over all queries, and (ii) specialized ranker trained on one of our groups versus a ranker trained on a potentially-diverse set of underperforming queries.

The group that we selected represented approximately 2% of the dissatisfaction instances, and had a *DSAT correlation* of 1.3. We selected this group because we were able to use existing query-document relevance judgments to quickly build a dataset with queries with document relevance judgments that belong to this group. Collecting query-document relevance judgments for thousands of queries and hundreds of documents per query is a costly and time consuming process. Crowdsourcing could be used to create more labeled datasets to target other DSAT groups.

We compare the results of three experiments. First, we use a general queryset to train a learning-to-rank algorithm. The general queryset contains queries that belong to the specific group among many other queries. Second, we train a specialized ranker using a queryset with queries that only belong to the selected group. If targeting specific underperforming groups is a good strategy, then the ranking function trained using a specialized query set will be better than the one using a general query set. Finally, to measure the value of coherence in our groups, we compare the performance of our specialized ranker with a ranker trained specifically for a set of poorly-performing queries, with low CTR.

The general ranker baseline is trained using a general queryset with 10,000 queries; some of them belong to the selected group. As noted above, the general ranker had access to attributes used in identifying DSAT groups, allowing it to learn to use DSAT information. Note that the general ranker is a state-of-the-art heavily optimized ranker using hundreds of features including clickthrough rate, link based features and content based features.

The specialized ranker is trained using a specialized dataset. The dataset contains queries that belong to the selected group only. The size of the specialized dataset is 2,000 queries. The size of the specialized dataset is much less than the size of the general dataset. This is not intentional and we believe that increasing the size of the specialized dataset could even lead to larger gains. The data used to train both rankers does not overlap with the data used to identify the underperforming query groups.

7.2.2 Effect of Specialized Ranker

We evaluated both the general and the specialized rankers using the specialized data using 10-fold cross validation. We evaluate the performance of both rankers using specialized data only because we are only altering the ranking on queries belonging to the selected group. The performance on all other queries will remain unchanged. The rationale behind this is that the all queries will go to the general ranker except for queries belonging to the selected group which will go to the specialized ranker. Note that these are not the same underperforming queries we identified earlier. Rather, they are new queries that belong to the underperforming group (i.e. all the attributes defining the group fire for them).

In Table 5, we report NDCG@1 through NDCG@5 improvements over the general ranker. Improvements are statistically significant at the 0.05 level according to the Wilcoxon *p-value*. The table shows that we observe a 1.52% NDCG@1 improvement and NDCG@5 improvement of 0.77%.

7.2.3 Effect of Group Coherence

We performed another experiment to measure the effect that the *coherence* of the identified dissatisfaction groups has in improving relevance when a specialized ranker is trained. We wanted to understand whether the relevance gains we achieved were specific to dissatisfaction groups, or if they can be obtained if we train a ranker on any set of poorly-performing queries. We construct a new data set with poorly performing queries (clickthrough rate less than 0.2), and with the same number of queries as the specialized dataset described earlier. We train a ranker using the same set of features on the low CTR dataset. We measure the performance using NDCG gain compared to the general ranker. The results are shown in Table 6. The results show that the gain is much smaller than the specialized ranker case and often not significant. The NDCG@1 gain is the only gain that is significant at the 0.05 level. This shows the value of the dissatisfaction groups and supports our hypothesis that group coherence is important for ranking.

8. DISCUSSION AND IMPLICATIONS

Identifying groups of queries where search engines underperform is an important research challenge. We have described a novel method to automatically identify dissatisfaction groups comprising queries where a search engine appears to be failing. For one such group identified by our method, we trained a specialized ranking algorithm and showed significant relevance gains over a state-of-the-art general ranker.

We validated our DSAT group identification approach by showing that groups identified were correlated with established measures of search engine performance, namely clickthrough rate and NDCG. However, unlike subsets of the queries identified by those metrics, queries were grouped so that they shared common attributes. As we showed through an experiment, such coherence is important for ranking. In particular, we showed that the specialized ranker performed better for our coherent groups than groups containing a broad range of queries that were just filtered based on a measure of retrieval effectiveness (low CTR in our case).

The methods presented combine to form an end-to-end framework capable of automatically identifying dissatisfied query groups and the adapting the ranking algorithm to them. However, the framework does not have to be used in its entirety. It may be desirable to first identify the groups of queries on which the search engine is underperforming and then triage those groups to make a determination regarding how to proceed. Deploying a specialized ranker may not always be appropriate depending on the nature of the group identified. For example, in DSAT groups where the searcher may benefit from interactive support rather than ranking improvements (e.g., a specialized direct answer on the SERP), interface modifications may be more appropriate.

We should also acknowledge some limitations of our study. The framework is limited by the need to gather dissatisfaction data before specialized rankers can be trained. New rankers that are released based on this data then need to wait some time (days or weeks) for new data to arrive. The method can only work on groups of sufficient size to provide a sufficient number of queries to train the ranker. This approach does not work for the smaller groups. One solution is to create group hierarchies to combine multiple smaller groups into a single large group based on shared attributes.

There are a number of important areas of future work. For the training and evaluation of our specialized ranking algorithms we used human relevance judgments. However, obtaining these judgments may be costly and we may not have such judgments readily available for the query groups identified by our algorithm. Prior work has shown that judgments may be obtained implicitly from search interaction [3][7] and we will explore the use of such judgments to further lessen the any remaining dependencies on human involvement in our framework (reducing additional costs where possible). We will also investigate the cost-benefit tradeoffs of adding more features tailored to the group of interest, rather than simply re-training the ranker for the current set of features. To handle smaller query groups for which we may lack sufficient training data to train a dedicated ranker, we will investigate how to combine multiple groups in a way that still maintains the group coherence that contributes to the success of our approach.

9. CONCLUSIONS AND FUTURE WORK

Search engines try to satisfy all users' needs and inevitably do not perform well for all queries. In this paper, we described and validated a method for automatic identification of dissatisfaction groups comprising queries where an engine is underperforming. Knowledge of these DSAT groups can help search engines since it can offer insights that inform engine design decisions and resource allocation. Importantly, rather than only identifying underperforming groups, we closed the loop and investigated automatically learning a specialized ranking algorithm that outperforms a general ranker with access to the same features. Our findings also show the benefit of group coherence in training specialized rankers. The method we describe in this paper can help search engines learn from evidence of searcher dissatisfaction and directly improve their search performance for troublesome queries. Future work will expand this research to target learning new ranking algorithms for multiple DSAT groups simultaneously and develop new methods to combine many smaller query groups to improve coverage.

REFERENCES

[1] Ageev, M., Guo, Q., Lagun, D., and Agichtein, E. (2011). Find it if you can: a game for modeling different types of web search success using interaction data. *Proc. SIGIR*, 345–354.

[2] Agichtein, E., Brill, E., and Dumais, S.T. (2006). Improving web search ranking by incorporating user behavior information. *Proc. SIGIR*, 19–26.

[3] Agichtein, E., Brill, E., Dumais, S.T., and Ragno, R. (2006). Learning user interaction models for predicting web search result preferences. *Proc. SIGIR*, 3–10.

[4] Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. *Proc. SIGMOD*, 207–216.

[5] Agrawal, R., and Srikant, R. (1994). Fast algorithms for mining association rules. *Proc. VLDB*, 487–499.

[6] Al-Maskari, A., Sanderson, M., and Clough, P. (2007). The relationship between IR effectiveness measures and user satisfaction. *Proc. SIGIR*, 773–774.

[7] Bennett, P.N., Radlinski, F., Yilmaz, E. and White, R.W. (2011). Inferring and using location metadata to personalize web search. *Proc. SIGIR*, 135–144.

[8] Carmel, D., Yom-Tov, E., Darlow, A., and Pelleg, D. (2006). What makes a query difficult? *Proc. SIGIR*, 390−397.

[9] Carterette, B. and Jones, R. (2007). Evaluating search engines by modeling the relationship between relevance and clicks. *Proc. NIPS*, 217–224.

[10] Cronen-Townsend, S., Zhou, Y., and Croft, W. B. (2002). Predicting query performance. *Proc. SIGIR*, 299−306.

[11] Dan, O., Dmitriev, P., and White, R.W. (2012). Mining for insights in the search engine query stream. *Proc. WWW*, 489-490.

[12] Downey, D., Dumais, S., and Horvitz, E. (2007). Model of searching and browsing: Languages, studies and applications. *Proc. IJCAI*, 2740–2747.

[13] Feild, H., Allan, J., and Jones, R. (2010). Predicting searcher frustration. *Proc. SIGIR*, 34−41.

[14] Fox, S., Karnawat, K., Mydland, M., Dumais, S.T., and White, T. (2005). Evaluating implicit measures to improve the search experience. *ACM TOIS*, 23(2): 147−168.

[15] Guo, Q., White, R.W., Dumais, S.T., Wang, J., and Anderson, B. (2010). Predicting query performance using query, result, and user interaction features. *Proc. RIAO*.

[16] Guo, Q., White, R.W., Zhang, Y., Anderson, B., and Dumais, S.T. (2011). Why searchers switch: understanding and predicting engine switching rationales. *Proc. SIGIR*. 335–344.

[17] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. (2009). The WEKA data mining software: an update, *SIGKDD Explorations*, 11(1).

[18] Han, J. and Kamber, M. (2006). Data Mining: Concepts and Techniques, Second Edition. *Morgan Kaufmann*.

[19] Han, J., Pei, J., Yin, Y., and Mao, R. (2004). Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Mining and Knowledge Discovery,* 8(1), 53–87.

[20] Hassan, A., Jones, R., and Klinkner, K.L. (2010). Beyond DCG: user behavior as a predictor of a successful search. *Proc. WSDM*, 221–230

[21] Hassan, A., Song, Y., and He, L. (2011). A task level user satisfaction model and its application on improving relevance estimation. *Proc. CIKM*, 125–134.

[22] Hauff, C., Murdock, V., and Baeza-Yates, R. (2008). Improved query difficulty prediction for the web. *Proc. CIKM*, 439–448.

[23] He, B. and Ounis, I. (2004). Inferring query performance using pre-retrieval predictors. *Proc. SPIRE*, 43–54.

[24] Hu, V., Stone, M., Pedersen, J., and White, R.W. (2011). Effects of search success on search engine re-use. *Proc. CIKM*, 1841−1846.

[25] Huffman, S. and Hochster, M. (2007). How well does result relevance predict session satisfaction? *Proc. SIGIR*, 567−574.

[26] K. Järvelin and Kekalainen. J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM TOIS*, 20(4), 422–446.

[27] K. Järvelin, S.L. Price, L.M.L. Delcambre, and M.L. Nielsen. (2008). Discounted cumulated gain based evaluation of multiple-query IR sessions. *Proc. ECIR*, 4–15.

[28] Dan Klein and Christopher D. Manning. (2003). Accurate unlexicalized parsing. *Proc. ACL*, 423–430

[29] Leskovec, J., Dumais, S., and Horvitz, E. (2007). Web projections: Learning from contextual subgraphs of the Web. *Proc. WWW*, 471–480.

[30] Li, H., Wang, Y., Zhang, D., Zhang, M., and Chang, E. (2008). PFP: parallel FP-growth for query recommendation. *Proc. RecSys*, 107–114.

[31] Teevan, J., Dumais, S., and Liebling, D. (2008). To personalize or not to personalize: Modeling queries with variation in user intent. *Proc. SIGIR*, 620−627.

[32] White, R.W. and Drucker, S.M. (2007). Investigating behavioral variability in Web search. *Proc. WWW*, 21−30.

[33] White, R.W. and Dumais, S. (2009). Characterizing and predicting search engine switching behavior. *Proc. CIKM*, 87−96.

[34] White, R.W. and Huang, J. (2010). Assessing the scenic route: measuring the value of search trails in web logs. *Proc. SIGIR*, 587–594.

[35] Xu, J., and Li, H. (2007). AdaRank: A boosting algorithm for information retrieval. *Proc. SIGIR*, 391−398.

[36] Zhou, Y. and Croft, W.B. (2007). Query performance prediction in Web search environments. *Proc. SIGIR*, 543−550.