Playing by the Rules: Mining Query Associations to Predict Search Performance

Youngho Kim[1]*, Ahmed Hassan[2], Ryen W. White[2], Yi-Min Wang[2]
[1] 140 Governors Drive, University of Massachusetts, Amherst, MA 01003, USA
[2] One Microsoft Way, Microsoft Research, Redmond, WA 98052, USA
yhkim@cs.umass.edu, {hassanam,ryenw,ymwang}@microsoft.com

ABSTRACT
Understanding the characteristics of queries where a search engine is failing is important for improving engine performance. Previous work largely relies on user-interaction features (e.g., clickthrough statistics) to identify such underperforming queries. However, relying on interaction behavior means that searchers need to become dissatisfied and need to exhibit that in their search behavior, by which point it may be too late to help them. In this paper, we propose a method to generate underperforming query identification rules instantly using topical and lexical attributes. The method first generates query attributes using sources such as topics, concepts (entities), and keywords in queries. Then, association rules are learned by exploiting the FP-growth algorithm and decision trees using underperforming query examples. We develop a query classification model capable of accurately estimating dissatisfaction using the generated rules, and demonstrate significant performance gains over state-of-the-art query performance prediction models.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Search Process

General Terms
Algorithms, Experimentation

Keywords
Underperforming query analysis; User dissatisfaction.

1. INTRODUCTION
Automatic detection of underperforming queries (where search engines are failing and users are dissatisfied with their search results) has been extensively studied [1][13][18][23]. In Web-search environments large volumes of query logs are readily attainable. This makes it affordable to collect many training examples that can be used to improve classification performance.

The ability to identify underperforming queries is especially important to Web search engines. Since those systems need to cover a broad range of diverse queries and since ranking algorithms are typically trained using a single sampled data set, there will be queries on which the ranking algorithms cannot execute effectively.

Previous work on predicting underperforming queries has primarily targeted user interactions (e.g., clickthrough statistics) [18][23] and combined features from user behaviors in query logs and physical sensors [13]. Although previous systems are quite successful, they are less effective from the search engine perspective. The reason is those dissatisfaction signals are only available *a posteriori* (e.g., once search is abandoned). This means that search engines rarely have the opportunity to interfere and remedy the user dissatisfied experience. Conversely, query performance predictors [10][24], for example, are more useful for that purpose because those predictors do not require interaction behavior signals and can help recognize dissatisfaction before the search session ends.

Table 1: Sample rule and example queries. Label indicates if query leads to satisfaction (SAT) or dissatisfaction (DSAT).

Association Rule		
{Movie, Art, "Robocop remake"} ⟹ {DSAT}		
No.	**Query**	**Label**
1	Robocop remake poster	DSAT
2	pics from Robocop remake	DSAT

In this paper, we propose a method for generating underperforming query identification rules using topical and lexical attributes of queries (e.g., topic of interest, important entities (concepts) in queries). We assume that there are frequent patterns among the features of underperforming queries and the association of those frequent features with dissatisfaction is helpful for identifying user queries likely to be associated with future dissatisfaction. Table 1 shows a sample association rule and examples of matching queries. For example, the rule {Movie, Art, "Robocop remake"} ⇒ {DSAT} consists of an antecedent with a set of binary attributes: Movie, Art, and "Robocop remake" and a consequent which is in our case a query satisfaction label: DSAT. Note that each query label is obtained using a DSAT prediction model [21], which is described briefly later. Since those attributes can be identified from only query, search systems can be informed of DSAT likelihood to the queries implied by the antecedent (LHS) of this rule promptly. Moreover, we can identify that current search systems cannot handle particular classes of queries successfully, which is valuable for improving the search experience.

In our system, given DSAT and SAT query examples, we perform two sub-tasks: (1) Attribute Generation, and (2) Association Rule Mining. In the first task, we define various attributes which can effectively characterize diverse Web queries. For example, categories in the Open Directory Project (ODP, dmoz.org) (e.g., Art) can be used as a binary attribute. Wikipedia categories and named entity categories (e.g., "Robocop"→ Movie) are also useful. Besides, we consider lexical attributes (e.g., "Robocop remake") because many DSAT queries frequently share some keywords. In addition to these topical attributes, we adopt behavior attributes that can characterize user-search actions (e.g., number of issued queries, and frequency

* This work was conducted while interning at Microsoft Research

WSDM'13, February 4–8, 2013, Rome, Italy.

of viewing search result pages). We extract distinct distributions of defined search actions, and label similar distributions for an identical category (i.e., a category label indicates a behavior attribute) by employing a spectral hashing algorithm [32]. From this, we can identify how behavior attributes (typically used in previous work) are combined with our topic attributes.

Using those attributes, we mine association rules to identify DSAT queries. Among many association rule learning techniques (e.g., [2][3]), we selected the FP-growth algorithm [20] because our system needs to handle a large amount of Web queries and FP-growth can effectively work on such large datasets [20]. In mining, we apply the FP-growth algorithm to both DSAT and SAT queries using categorical attributes (e.g., ODP categories), and identify what subsets of the attributes (item-sets) frequently appear in DSAT queries. In other words, the identified item-sets (antecedent) imply DSAT queries (i.e., the consequent of the rule is {DSAT}). After generating categorical rules, we augment the antecedent of the rules with lexical attributes. Since the number of lexical attributes (i.e., keywords) extracted from all queries is prohibitively large, we adopt a two-tiered approach where we first extract categorical rules and then mine lexical rules using queries to which certain categorical rules apply.

We show the effectiveness of our rule mining system by verifying that the system is capable of generating effective DSAT segments. We also demonstrate the effectiveness of the rules by using them as features for building a DSAT prediction system and show that it achieves superior performance when compared to state-of-the-art query performance prediction baselines [10][17][25].

The remainder of this paper is structured as follows. Section 2 describes relevant related work in the areas of search satisfaction, search quality analysis, and query performance. Section 3 defines our problem and Section 4 describes the main approach. We present experimental results in Section 5 and conclude in Section 6.

2. RELATED WORK

Relevant research in a number of areas is presented: (1) satisfaction, (2) search quality, and (3) query performance prediction.

2.1 Search Satisfaction

There has been significant prior work on deriving signals related to search satisfaction and success from online behavior. Methods for doing this typically correlate users' search behavior with their in-situ self-reporting [16] or judgments of search success provided by expert judges [18][22]. Early investigations by Fox et al. [16] correlated self-reported measures of search satisfaction with interaction signals gathered implicitly from search behavior, such as search-result clicks and dwell time for clicks. They deployed an instrumented browser and showed a relationship between explicit measures of search satisfaction and implicit measures derived from search behavior. In particular, they found that short dwell times and clicks on many results for a query were both indicators of search dissatisfaction. Ageev et al. [1] propose a formalization of different types of success for informational search via a scalable game-like infrastructure for crowdsourcing search behavior studies. They show that their model can predict search success effectively on their data and on a separate set of logs comprising search engine sessions. Hassan et al. [23] 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. In recent follow-up work, Hassan [21] proposed a semi-supervised approach for search satisfaction modeling via both labeled and unlabeled data.

2.2 Search Quality Analysis

Beyond studying search session satisfaction in isolation, others have investigated how the quality of search results can affect search behavior. Huffman and Hochster [26] discovered a fairly strong correlation with search 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. Aula et al. [6] investigated the behavioral signals that are suggestive of a user struggling in a search task. They showed in a laboratory study and in a larger-scale remote study that when searchers have difficulty in finding relevant information, they formulate more diverse queries, they are more likely to use advanced operators (e.g., '+', '–', 'OR'), and they spend a longer time on the search result page as compared to the successful tasks. Feild et al. [13] studied the affective impact of searching and developed methods to predict search frustration from behavioral and physiological signals. They gave users difficult information seeking tasks and estimated 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. [18] characterized the reasons that searchers switch via a browser plugin that captured an explanation at switch time. They showed that one of the primary reasons that people switched was dissatisfaction with the search results on the pre-switch engine.

2.3 Query Performance

Search engine performance for a particular query is typically measured using relevance metrics such as precision and recall. Al-Maskari et al. [4] found a reasonable correlation between many information retrieval metrics and satisfaction with result rankings. Beyond performance measurement, research on *predicting* query performance has been conducted to understand differences in the quality of search results provided by search systems for different queries. Such predictions do not require relevance judgments (at least not when the models are being applied, but perhaps during a separate training phase) and can be used to determine when to use additional computational resources or use alternative methods (e.g., specialized ranking algorithms or different interface support) to improve results for difficult queries. While it has been shown that different query representations [8] or retrieval models [9] improve search performance, it is more challenging to accurately predict which methods to use for a particular query.

Methods using Jensen-Shannon divergence [8], query clarity [10], and weighted information gain [35] have been developed to predict the retrieval performance on a query (e.g., as measured via precision) post retrieval. Rather than using post-retrieval query-document relevance scores, which can take time to compute, He and Ounis [24] proposed the use of pre-retrieval properties that can be generated prior to the retrieval process (e.g., query length, query scope). Pre-retrieval predictors are advantageous because they can be calculated during indexing, rather than waiting for the query to be received, making them more efficient. Zhao et al. [33] propose pre-retrieval predictors based on the similarity between a query and the underlying collection and the variability with which query terms occur in documents. Leskovec et al. [28] used graphical properties of the link structure of the result set to predict the quality of the set and the likelihood of subsequent query reformulation. Some research has also been conducted on predicting query performance using searcher interaction behavior. Carterette and Jones [9] used click-through behavior to evaluate the quality of search advertising results, but they did not study other user interaction features, and their focus was on search advertising not general Web search. Guo

et al. [17] used behavioral features, including engine switches, to predict query performance. Their method involved the use of interaction logs for a large number of their most important prediction features. However, the reliance on log data limits the generalizability of the model, meaning that it can only be applied in commercial search engines and evaluating the model outside of those settings is potentially problematic.

Our work extends the research presented in this section in a number of ways. First, rather than simply modeling or identifying the attributes of search satisfaction, we use apply DSAT modeling to identify queries where the search engine may be underperforming. Second, the method that we propose is not dependent on interaction behavior meaning that our approach can help before users have to experience dissatisfaction with the engine. Also, by not relying on interaction behavior makes our results more reproducible by academic researchers trying make advances in query performance prediction. Third, we demonstrate through a large-scale evaluation with search queries drawn from search logs that our approach outperforms several baselines that draw on query properties, search interaction, and their combination.

3. PROBLEM FORMULATION

In this section, we formulate the task of mining DSAT association rules and predicting query performance. We start by defining terms that we will use throughout the paper.

Definition 1 (Search Session) A *search session* is a sequence of user actions that begin with a query, includes subsequent queries, URL visits, click information, and ends with a period of inactivity. We assume that a session ends if the user was idle on a page for over 30 minutes, typically used in previous work, e.g., [12].

Definition 2 (Search Goal) A *search goal* is a single information need that may result in one or more queries. Every search session could be segmented into one or more search goals. We adopt the search goal definition and the goal extraction method from [27]. Using the prediction model in [21], we label each goal to either of {DSAT, SAT} and use a search goal as an instance in mining. We assume that each instance (i.e., search goal) is represented by the first query. If we use the subsequent queries of an instance, we explicitly specify "subsequent" queries.

Definition 3 (Attribute): An *attribute* is a binary property of a query instance that describes a specific characteristic of that instance. For example, the attribute "Shopping" indicates that the query is topically related to retail.

In this work, we present solutions to the following problems:

Problem 1 (Attribute Generation): *Attribute generation* is the process of defining and generating a set of attributes to represent query instances. Given a query instance, our objective is to generate effective attributes and associate (binary) values to them. The attributes should describe different aspects of the query instance and they should be dynamic in the sense that they may be easily extendible and need minimal human intervention to define.

Problem 2 (DSAT Association Rule Mining): *DSAT association rule mining* is the process of discovering frequent patterns (subsets) of generated attributes (from Problem 1) to identify DSAT queries. Given attributes and labeled queries, we generate rules that imply DSAT. The task is formally defined as follows.

Let $A = \{a_1, a_2, \dots, a_n\}$ be a set of n binary attributes and $Q = \{q_1, q_2, \dots, q_m\}$ be a set of (query) instances where each instance is labeled as DSAT or SAT. Based on A, each instance can be represented by a vector of binary attributes, e.g., $q_i = \{a_1 = 1, a_2 = 0, \dots, a_n = 1\}$. Then, an association rule can be formulated as $\{a_i, a_j, \dots\} \Rightarrow \{\text{DSAT}\}$ where a_i and a_j in LHS are the frequent attributes of DSAT instances, and as an output, we generate multiple association rules. Since we are interested in rules whose consequent (RHS) is {DSAT}, we simply represent every rule by only its antecedent (LHS). The instances represented by a rule indicate queries implied by the antecedent of the rule. Based on mined rules, DSAT segments are formed by the queries matched to the rules.

Problem 3 (DSAT Prediction): *DSAT Prediction* is the process of predicting, at query time, whether a query will be satisfied or not. In solving this, we use the association rules defined in Problem 2.

4. MINING UNDERPERFORMING QUERY ASSOCIATIONS

We now present our methodology for generating underperforming (DSAT) query identification rules. We first describe how useful attributes are defined for rule generation, and then explain how to generate DSAT identification rules using the defined attributes.

4.1 Attribute Generation

Generally, many attributes can be used to describe query impressions including query topic, query entities, and search behavior. However, most behavior-related features (e.g., search state [1], dwelling time [23]) have been examined in previous studies, and behavior information is typically available only after the user has abandoned searches. Instead, we attempt to devise topical and lexical attributes (which can be mainly identified from query texts) and the values for these attributes are instantly computed from current queries. For this, we exploit the categories in ODP, Named Entity Recognition (NER) [30], and Wikipedia (wikipedia.org). In addition, we also generate behavior attributes from distributions of historic user search actions. Thus, given a query instance, we generate attributes described as follows.

4.1.1 ODP Attributes

Open Directory Project (ODP, dmoz.org) is a hierarchical ontology for Web pages and lists similar topic pages in the same category including smaller categories. We use each ODP category as a binary attribute, and the ODP classifier, proposed in [7], can classify queries into 219 categories from the top two levels of the ODP hierarchy (see [7] for more details, including results of a performance evaluation). Specifically, for each query instance, the classifier outputs top-5 probable categories, and among them we select the categories whose probabilities are greater than 0.5 (empirically set) as attributes. In our experiments, we allow more than one category to describe every query instance.

4.1.2 Named Entity Attributes

Named Entity Recognition (NER) is the process of identifying entities (e.g., $People$, $Locations$, $Organizations$, etc.) from text. NER systems are oftentimes domain dependent and need large amounts of labeled data for training. This makes them less suitable for Web search queries that are short in nature and not usually represented by well-formed sentences. For our purpose we need a named entity tagger that can be easily adapted to new classes, works well on short and ill-formed text, and uses available knowledge sources instead of domain-dependent human labeled training data.

We rely on Wikipedia and n-gram Language Models (LMs) to tag named entities in queries [30]. Specifically, given a sequence of words, salient entities in a domain-specific LM are identified. A

User	Q	SERP	SR	SR_long	Ad
A	4	1	1	0	0
B	2	4	2	2	0

Figure 1: Sample distributions of search action frequencies. Q, SERP, SR, SR_long, and Ad indicate a query submission, search result page, search result click, long dwell (> 30s) result click, and ad click, respectively.

domain-specific LM is generated by a domain corpus (e.g., "Movies" LM contains n-gram statistics of a movie corpus). Salient domain entities are identified by measuring statistical differences between the domain model (foreground) and general LM (background). For each word, if a word belongs to the background, or the outside-tag, the word should be generated using the background LM. On the other hand, if the word belongs to the foreground, begin-tag, or inside-tag, the word should be generated using the foreground LM that named entities are built from. Weak unsupervised signals from each domain corpus (e.g., Wikipedia titles and click counts) are used to estimate parameters in the recognition model. This approach achieves 0.51 precision and 0.48 recall [30], when applied to short and not necessarily grammatical dataset with 1,300 queries, compared to 0.35 precision and 0.43 recall from a Conditional Random Fields system [15] .

One motivation to use this NER tagger is that topics in Web queries are diverse and recognizing only basic entities (i.e., Person, Location, and Organization) is less useful for identifying effective attributes. Thus, to cover various web topics, we train the classifier using domain language models corresponding to the following domains (*Person, Location, Organization, Food, Restaurant, Local review, Wikipedia, Movie,* and *Game*). We use each domain (category) as a binary attribute, and assume that a query can contain multiple attributes if the query contains entities from multiple domains. As an example, "Caesar salad recipe in TGI Friday's" can contain the attributes $\{Food, Restaurant\}$ because "Caesar salad" is a $Food$ entity and "TGI Friday's" is a $Restaurant$ entity.

4.1.3 Wikipedia Attributes

Wikipedia is an online encyclopedia which contains four million articles (entities). We use Wikipedia entities to generate a set of descriptive and diverse attributes. Specifically, we first crawl about one million Wikipedia articles, and extract Wikipedia entities from query texts as described in the previous subsection. Then, for each extracted entity, we identify the most relevant Wikipedia article by using a surface-level exact matching. (Note that other types of matching (e.g., semantic-level) can be used further if it can perform reasonably well). Once Wikipedia articles are matched to the query entities, we extract Wikipedia topic-categories from each article. Each Wikipedia article generally contains one or more topic-categories, e.g., the article of "Ford Mustang" includes Ford Vehicle, Coupe, etc. As a result, we can link a query instance to the categories via identified entities from the query texts, and use those categories as attributes. In experiments, each query instance contains 0.9 entities on average, and 4,482 Wikipedia topic-categories are extracted and used as attributes.

4.1.4 Behavior Attributes

Here we present a method to generate behavior attributes. Since previous work proves that using behavior information is effective to predict query performance [17], we devise the attributes which recognize the characteristics of the general search behavior of users. We assume that dissatisfied users act differently in searches compared to satisfied users. For example, we observed that some dissatisfied users issue many queries, but spend less time on examining search results. To capture such behavior characteristics, we generate a distribution of search action frequencies from each query instance. Specifically, we define seven distinct search actions and identify frequent distributions that occur more than 10 times (i.e., we ignore extremely rare patterns). Figure 1 shows some examples of the distributions. Then, we group similar distributions into the same cluster (attribute) by using spectral hashing [32], which can develop a hashing function that maps similar vectors into the same bucket (i.e., hash code). Given a set of behavior distributions, we consider a distribution as an input vector and run the spectral hashing algorithm [32] with *k*-bits (the bit length of a hashing code). After a hash function is obtained, we use a mapped hash code as a binary attribute. In experiments, we hash 3,069 distinct distributions into 2-bit codes (i.e., four different hash values are mapped), and accordingly four behavior attributes are generated.

Table 2: Mean frequencies of search actions. Due to the space limit, the statistics of less significant actions (e.g., Ad) are omitted. In each column, a superscript indicates a significant difference at $p < 0.01$ using the Wilcoxon rank-sum test, e.g., ^{C13} denotes a significant difference from the code of 1 and 3.

Code	Q	SERP	SR	SR_long
0	5.01^{C13}	0.18	0.38	0.56
1	1.60	0.53	0.56	0.61
2	4.82^{C13}	1.18^{C01}	1.09^{C01}	1.55^{C01}
3	1.74	1.96^{C01}	1.26^{C01}	1.35^{C01}

One motivation for using spectral hashing is that a hashing-based algorithm is much faster than clustering algorithms, such as k-Nearest Neighbor, in learning patterns. Moreover, in pilot experiments, a simple clustering method failed to find a reasonable number of clusters (attributes); only a single cluster was identified. To understand the effectiveness of using the spectral hashing, we conduct pilot experiments using about 140,000 session instances randomly sampled from our dataset. From that data, we extract 237 distinct distributions of search action frequencies and generate 4-bit hash codes. Then, as shown in Table 2, we measure the mean frequency of each action among the instances mapped to each code (i.e., 0, 1, 2, and 3). First, the instances mapped to 0 and 2 include significantly more query submission (Q) while the instances belonging to 1 and 3 have fewer query submissions. Second, regarding search result actions (i.e., SERP, SR, and SR_long), the users in 2 and 3 spend more times on SERP and SR than the users in 0 and 1. To prove these, we perform a statistical significance test on each of those actions (see Table 2). In summary, we can characterize the users (instances) in each code; 0 indicates more query submission and short search result examination, 1 denotes less query submission and short search result examination (generally most DSAT users belong to this category), 2 indicates more query submission and long search result examination, and 3 indicates relatively less query submission and long search result examination (i.e., spend more time on a few search results).

4.2 DSAT Association Rule Mining

In this section, we describe our method for finding DSAT association rules using FP-growth algorithm and decision trees. Specifically, we first use the FP-growth algorithm with the "categorical" attributes (i.e., ODP, NER, Wikipedia, and Behavior) to identify categorical rules (e.g., {Movie, Art}). After this, we apply Decision Tree learning using "lexical" attributes that include n-gram keywords from the queries identified by the categorical rules, and extract lexical rules (e.g., {Movie, Art, "Robocop remake"}). We provide the details of each method in the remainder of this section.

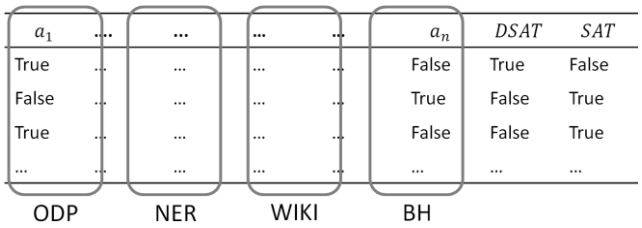

Figure 2: Input data set for FP-growth algorithm.

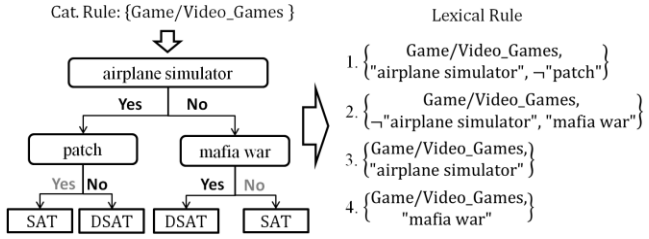

Figure 3: Decision tree-based lexical rule generation.

4.2.1 Categorical Rule Mining

Given a set of attributes, we identify subsets of attributes (item-sets) to identify DSAT instances. For this, we can utilize several well-known approaches to find frequent patterns in transactional or relational data sets that are described in the data-mining literature. A typical example of this approach is the market basket analysis problem [2]. Using a similar formulation, we formally define the DSAT association rule mining problem as follows: Let $A = \{a_1, a_2, \dots, a_n\}$ be a set of n binary attributes containing ODP, NER, Wikipedia (WIKI), and Behavior (BH) attributes, and $Q = \{q_1, q_2, \dots, q_m\}$ be a set of instances and each instance is labeled as DSAT or SAT. We define a new attribute set, $A' = \{a_1, a_2, \dots, a_n\} \cup \{\text{DSAT}, \text{SAT}\}$ where the labels are included as attributes. Figure 2 illustrate an input data set, $\{I \times A'\}$. Then, a "DSAT" association rule is defined as an implication of the form $\text{X} \Rightarrow \text{Y}$ where $\text{X} \subseteq A$, $\text{Y} = \{\text{DSAT}\}$ and $\text{X} \cap \text{Y} = \emptyset$.

Among many algorithms to solve this problem (e.g., [3]), we chose to use the FP-growth algorithm [20] because it is more efficient than generate-and-test algorithms given that it adopts a divide-and-conquer strategy. The proposed approach should be able to handle millions of instances and hence efficient rule mining is necessary. Briefly, the algorithm consists of two main steps. First, it builds the FP-tree which efficiently represents information about item-set (subset of attributes) association, and in the next step, frequent patterns are mined from the FP-tree. To build the tree, the whole data set is scanned twice; in the first scan, the attributes are sorted in descending order by their frequencies, and the tree is constructed by spanning attribute nodes from more frequent attributes to less frequent ones, scanning each row in the data (see [20] for details).

Before running the algorithm, to reduce the computational complexity and obtain more effective rules, we define 3 constraints:

1) The number of attributes from a single group may not exceed a threshold l.
2) Minimum support-level needs to be reasonably small.
3) Minimum confidence-level is more than or equal to the portion of DSAT instances to the whole data set.

In our data set, originally there are four different attribute groups (i.e., ODP, NER, WIKI, BH), and the first constraint limits a mining path (from the root to a leaf node in the FP-tree) to have maximally $4l$ different attributes from those groups. In other words, a mined rule resembles any combination of {[ODP], [NER], [WIKI], [BH]} where each attribute group (bracket) contains at most l attributes from the corresponding group. Thus, we impose a constraint on the maximum depth of the tree, which can reduce the time-complexity in practice. Since the algorithm for mining the FP-tree structure is a recursive procedure during which many sub FP-trees are created, too long paths (e.g., containing more than 100 nodes (attributes)) are problematic and defining an effective length is complicated. The second and the third constraints allows more effective rules to be generated. The support (S) of an item-set (X) is defined as the proportion of instances in the data set which contain X, and the minimum support level denotes the minimum number of instances which support a generated rule. If the rule can cover many queries (instances), it may contain more SAT queries and is ineffective. Thus, targeting relatively small number of instances is more helpful to obtain effective rules, and in experiments, we have about one million instances and we set S to 20.

Based on S, the confidence (C) of a rule is defined as $C(\text{X} \Longrightarrow \text{Y}) = S(\text{X} \cup \text{Y})/S(\text{X})$, an estimation of probability $\text{p}(\text{Y}|\text{X})$. By the third constraint, we can recognize effective rules which can cover relatively more DSAT queries comparing to the general proportion of DSAT instances to the whole dataset, $\text{p}(\text{DSAT}|\text{X}) \geq \text{p}(\text{DSAT})$. However, a confidence level that is too high can also mean that very few rules are extracted because generally $\text{p}(\text{DSAT})$ is low, which is also ineffective.

Next, we run the FP-growth algorithm on the augmented data set with two parameters, S and C. We use an improved-version of FP-growth implemented by [11], which can reorder and prune the input data so that more common attributes appear first and unsupported 1-item-sets are discarded. Among the extracted rules, we only select the ones whose consequent (RHS) is DSAT (i.e., $\text{Y} = \{\text{DSAT}\}$). As a result of this step, we generate multiple categorical rules that contain effective attributes for identifying DSAT queries.

4.2.2 Lexical Rule Mining

In this section, we provide the details of generating lexical rules that use DSAT query keywords as lexical attributes.

One motivation for adopting lexical rules is to identify more specific information about DSAT queries. Although categorical attributes provide general topic information of DSAT queries (e.g., search engines perform worse with queries related to the categorical attributes, Ford Vehicle and Shopping/Vehicle), identifying detailed lexical information (e.g., among Ford Vehicle and Shopping/Vehicles queries, the queries containing "lease" are not properly handled) would be much more helpful for analysis purposes. Besides, more specific rules are more likely to have higher confidence (i.e., $\text{p}(\text{DSAT}|\text{X})$) than general rules because typically $\text{p}(\text{DSAT})$ is much lower than $\text{p}(\text{SAT})$.

We generate a lexical rule by combining categorical rules with discriminative keywords in $DSAT$ queries. The formal definition is given as: $\text{X} \cup \text{Z} \Longrightarrow \text{Y}$ where $\text{X} \subseteq A$, $\text{Z} \subseteq K$, $\text{Y} = \{\text{DSAT}\}$, $(\text{X} \cup \text{Z}) \cap \text{Y} = \emptyset$, A is the attribute set (Section 4.1), and K is the set of keywords extracted from the DSAT queries represented by X. Since effective categorical rules (i.e., $\text{X} \Longrightarrow \text{Y}$) are already extracted, we need to identify effective DSAT keywords (i.e., Z).

To solve this problem, we propose a decision tree learning-based method which extracts effective lexical attributes from DSAT query texts. If we could train a decision tree where a node corresponds to a term appearing in DSAT queries (instances) in order to determine whether a query is DSAT, a decision rule which represents DSAT examples includes a set of discriminative terms (features). In addition, to combine those key terms with categorical rules, for each

categorical rule, we identify the set of queries implied by the categorical rule, and generate a decision tree using the instance set. Then, extracted keywords are effective when combined with the categorical rule. Figure 3 depicts how lexical rules are generated from a categorical rule. Since decision trees can represent negation, we originally extract the first and second lexical rules containing negated lexical attributes (e.g., ¬"patch"). However, to generalize these rules (i.e., expanding their coverage in real situations), we can ignore negation and the third and fourth rules can be formulated. Figure 4 describes the algorithm.

For the lexical rule generation, we can alternatively consider pattern mining approaches (e.g., FP-growth algorithm [20] used in Section 4.2.1) using keywords as attributes. However, in comparison with the FP-growth algorithm, decision tree learning has the following advantage: decision tree rules can imply negation whereas FP-growth algorithm mines only association rules that do not imply negation. In some circumstances, negation is important because some attributes rarely appear in DSAT queries but frequently appear in SAT queries. For example, let us consider the first lexical rule in Figure 3. This rule indicates that among the queries represented by {Game/Video_Games, "airplane simulator"}, the queries not related to "patch" are DSAT (otherwise the queries are SAT), i.e., not every query in {Game/Video_Games, "airplane simulator"} is DSAT. Moreover, if we ignore negation (e.g., the third and fourth rules in Figure 3), the mining results are almost identical to the ones identified by the FP-growth algorithm (ideally, checking the confidence of each decision tree rule can be necessary, but in our experiments, every decision rule ignoring negation had $p(\text{DSAT}|\text{X}) \geq p(\text{DSAT})$) because the FP-growth algorithm uses the confidence (the proportion of DSAT to the instances represented by the rule) to identify DSAT association rules and similarly decision-tree learning also considers the entropy of each attribute (the proportion of DSAT to the examples that contain selected attributes) in generating DSAT decision rules. In our experiments, we discovered that the rules including negation can provide more specific information. We describe some examples in Section 5.3. We use WEKA implementation of C4.5 algorithm [19] to train decision trees, query texts are stemmed by WordNet [14], and stop-words are removed.

4.3 DSAT Query Prediction Model

In this section, we describe a DSAT query prediction model that can classify a given query into DSAT or SAT. Since we generated categorical and lexical rules to be effective to identify DSAT queries (Section 4.2) we use these rules as features, and train the model using labeled examples. The formal definition is given as follows.

Suppose that $R = \{r_1, r_2, \dots, r_n\}$ is a set of n rules, $Q = \{q_1, q_2, \dots, q_m\}$ is a set of m query examples, and the label of each example $L(q_i)$ is known. For training, a feature vector of each q_i is created as $x_{q_i} = \{r_1(q_i), r_2(q_i), \dots, r_n(q_i)\}$ where $r_j(q_i)$ is a binary value;

if q_i is matched to r_j then 1; otherwise, 0. Then, a set of training examples is given as $\text{X} = \{\langle x_{q_i}, L(q_i)\rangle\}_{i=1}^{m}$, and a classification function $f: \text{X} \mapsto \{1,0\}$ maps a feature vector associated with a query to a binary label where 1 and 0 indicate DSAT and SAT, respectively, and the model is learned to minimize a loss function defined by the disagreement between a mapped label and original label, $L(q_i)$ for every training example. For learning, we use Support Vector Machine and Logistic Regression (see Section 5.2).

ALGORITHM Lexical Rule Generation
INPUT:
- Set of labeled query instances, $Q = \{q_1, q_2, \dots, q_n\}$ where the label of q is $DSAT$ or SAT, i.e., $L(q) \in \{\text{DSAT}, \text{SAT}\}$
- Categorical rule, C: X $\Longrightarrow$ {DSAT} where X is a subset of categorical attributes (item-set)
- Boolean value, b, which indicates ignoring negation

OUTPUT: A set of lexical rules
PROCESS:
1) Initialize $R = \{\ \}$
2) Find the set of query instances implied by X, i.e., $Q' = \{q|\text{X}(q) = 1\}$ and $q \in Q$
3) If $Q' = \emptyset$ Then return $\emptyset$
4) Generate K which contains n-grams extracted from Q'
5) Train a binary decision tree, DT, which uses K as features and Q' as examples where $L(q)$ indicates a label of $q \in Q'$
6) For each $DSAT$ leaf node, l_{DSAT}
   a. Generate Z the set of nodes (features) in the path from the root to l_{DSAT} in DT
   b. If $b = 1$ Then exclude the node corresponding to negation from Z
   c. Append Z into R
7) End For
8) Return $C \times R$

Figure 2: Lexical rule generation algorithm.

5. EXPERIMENTS

In this section, we provide experimental results of our DSAT identification rule mining system. In Section 5.1, we analyze the generated rules in terms of topical cohesiveness, clickthrough rate, and DSAT correlation. Section 5.2 describes our query performance classification model that uses the mined rules to predict dissatisfaction of queries, and demonstrates its effectiveness.

5.1 Rule Analysis

5.1.1 Experimental Set-up

We obtained interaction logs (from June 2012) for 1.5 million search sessions from hundreds of thousands of consenting users using several commercial search engines 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.

Using the DSAT prediction model [21], we classify each instance into either of DSAT or SAT. To implement [21], we develop a semi-supervised model; we first build a supervised classifier using a small set of labeled DSAT and SAT query instances, and by this classifier, initial parameters are obtained; the membership of unlabeled data is calculated using an updated model whose parameters are re-estimated by the Expectation-Maximum algorithm. Then, we randomly split the data into two distinct equally-sized groups, non-overlapping in time: *mining* and *analysis*. We use the mining set to generate association rules (setting the support (S) to 20), and analyze rules using the analysis set (i.e., unseen data).

In analysis, we use three different metrics defined as follows.

Table 3: Basic rule statistics. OUR includes both categorical and lexical rules. The percentile ratio indicates the reuse ratio of the generated rules in the analysis set.

Metric \ Method	TC (baseline)	OUR
# of rules generated from *mining set*	1,160	26,845
# of rules matched in *analysis set*	1,158 (*99.83*%)	5,861 (*21.83*%)
Percentage of the matched queries in the *analysis set*	0.7310	0.7520

(**Cosine-Similarity**) Cosine-Similarity is measured to identify the topical similarity between the instances (queries) represented by each rule. We hypothesize that more topically cohesive segment (set of queries) would be obtained if the rule is effective. We describe each query by a vector of all unique terms in the whole data set and each dimension corresponds to the frequency of a term in the query. The similarity between two queries, q_i and q_j is given:

$$\text{similarity} = \frac{q_i \cdot q_j}{\|q_i\|\|q_j\|}$$

(**Clickthrough Rate**) Clickthrough Rate (CTR) indicates the fraction of times the query results in a click on the algorithmic results when it is issued. Given a query, CTR is computed by:

$$\text{CTR} = \frac{\#\text{ of algorithmic clicks}}{\#\text{ of impressions}}$$

CTR is calculated using a different and much bigger set of queries that does not overlap with our data set.

(**DSAT Correlation**) DSAT Correlation is a simple measure that estimates the correlation/dependence between a rule and dissatisfaction. Given a rule, $X \Longrightarrow \{\text{DSAT}\}$, the metric is defined as:

$$\text{DSAT Correlation} = \frac{\text{p}(X, \text{DSAT})}{\text{p}(X)\text{p}(\text{DSAT})} = \frac{\text{p}(\text{DSAT}|X)}{\text{p}(\text{DSAT})}$$

$\text{p}(\text{DSAT}|X)$ is the confidence of X defined in Section 4.2.1, and $\text{p}(\text{DSAT})$ is estimated by the proportion of DSAT instances in the data. If the value is exactly 1, X and dissatisfaction is independent, and if the value is greater than 1, X is positively correlated with dissatisfaction. Otherwise, X is negatively correlated with DSAT. Comparing with the confidence, this metric is more robust because the confidence can be misleading if the data are imbalanced, and generally much more SAT queries are recognized in query logs. In addition, we considered the confidence in rule generation.

(**Fixed Attribute Set Baseline**) Traditionally, some predefined query attributes (e.g., IsCommerce indicates the query is related "Commerce" topics) have been used to segment queries into classes. For this, text classifiers are trained using manually labeled data. We refer to those attributes as the Text Classification (TC) attributes. As a baseline, we generate association rules using these query attributes, and compare the rules generated by our system to the TC rules. In our method, we do not consider using the TC attributes because they need extensive human involvement in terms of selecting which classifiers to build and labeling training data. In addition to the need for extensive human involvement, using these attributes has two more downsides. First, they put a burden on the human building the system to decide which attributes to consider and hence which classifiers to build. Contrast this with the proposed approach where a huge number of attributes is automatically introduced using existing resources (e.g. Wikipedia). Second, these classifiers may be readily available to commercial search engines as private properties, but they are not publicly available which limits their usefulness to the research community. Instead, we use them as a baseline to validate our method which utilizes public resources. For generating TC rules, we use a set of 60 proprietary classifiers (e.g. Technology, Travel, etc.) and we apply the FP-growth algorithm to the same data set (*mining set*).

Table 4: Comparison against TC baseline. Bold indicates statistically significant difference at $p < 0.01$ from TC (using Wilcoxon rank-sum test).

Metric \ Method	TC	OUR
Avg. Cosine-Similarity	0.0133	**0.5296**
Avg. CTR	0.4509	0.4554
Avg. DSAT Correlation	1.9420	**4.0141**
Avg. # of queries	2122.44	**530.79**
Correlation Coefficient (CTR vs. DSAT Correlation)	−0.2718	−0.3204

5.1.2 Analysis Results

Table 3 shows basic statistics of rule generation using our system and the baseline. Using the mining set, we generate lexical rules with ignoring negation (Section 4.2.2) because negation can reduce the rule coverage and the rules containing negation are hard to reuse in unseen data (*analysis set*). As a result, 3,859 categorical rules and 22,986 lexical rules are generated (OUR). For TC, 1,160 rules are extracted from the same data. To identify rule coverage, we measure the number of the rules that can match any analysis instances, i.e., reusable with unseen data. Comparing with TC, our system can generate many more rules but many of them are not reusable in unseen data. That said, slightly more examples in the analysis set can be represented by the rules generated by our system, i.e., generally both systems can cover the almost same amount of new queries.

Next we provide the results using the metrics defined in Section 5.1.1. For each rule, we first identify which analysis instances can be matched (i.e., generate clusters), and calculate the average statistic of each metric in a cluster. Every rule corresponds to a query cluster which contains all queries that match this rule. Then, we report the mean of the average value of each metric over all clusters. Table 4 shows the results of each metric. First, the clusters obtained by OUR rules are more topically coherent. Since lexical rules containing keywords of DSAT queries are quite topically specific, this result is somewhat straightforward. Second, the rules generated by our system are strongly positively correlated with dissatisfaction, which means that OUR rules are more effective for identifying DSAT queries than TC rules. Third, the CTR of both systems is not significantly different. Generally, low CTR indicates dissatisfaction, but sometimes DSAT queries can include more clicks. For example, users do more clicks on search results but they immediately leave from the clicked results because those are not relevant. However, DSAT Correlation provides more direct relation between the rules and DSAT. Lastly, we calculate Kendall's τ coefficient between avg. CTR and DSAT Correlation. In both systems, DSAT Correlation is negatively correlated with CTR, which means generally low CTR indicates dissatisfaction. Besides, more salient negative relation is identified by OUR as is evidenced by a much higher DSAT Correlation in OUR.

5.2 DSAT Query Classification

5.2.1 Experimental Set-up

We conduct experiments to evaluate our system in DSAT query classification (Section 4.3). In this, a system would perform better if its features (e.g., association rules in our system) are more effective to identify DSAT queries. We use the mining set to generate the

Table 5: Summary of Baseline Features.

Baseline	Class	Feature
Baseline 1	Query	query clarity score [10] query word length inverse web term probability [25]
Baseline 2 [17]	Query	query word length query character length
	User Interaction	# of distinct subsequent queries, # of SR, # of SERP, # of all search actions, # of clicked answers query impression count, # of long actions (dwelling time is more than 30 seconds), # of algorithmic clicks, clickthrough rate (CTR), $\text{SAT Rate} = \frac{\#\text{ of long actions}}{\#\text{ of algorithm clicks}}$

rules, and exclude all behavior attributes because we are interested in predicting query performance at query time before observing any user behavior. For evaluation, a balanced data set is used, which contains 40,000 query instances (i.e., 20,000 instances are DSAT and the others are SAT); this comes from another random sampling of the queries that does not overlap with the mining set. For learning, we use a Linear Support Vector Machine (SVM) [29] and Logistic Regression (LR) [5]. We run each classifier 10 times and for each run, we perform 10-fold cross-validation using random partitioning.

5.2.2 Baselines

To develop baselines, we leverage the features from previous work on predicting query performance [10][17][25]. We compare our system to four different baselines each of which uses different features. Table 5 summarizes the features used in Baseline 1 and 2.

The first baseline is developed by using a combination of the query performance predictors presented in [10][25]. We chose to use query clarity score ([10]) because it performs as well as other models (e.g. [34]), does not require access to external resources ([34] requires the ranking results from two different search algorithms), and is not limited to any specific query types ([34] focuses on only two types of queries; named-page finding and content-based). To estimate inverse collection term frequency ([25]), we used term probabilities obtained from the Web N-Gram services [31] that provide smoothed n-gram probability. We calculate the sum, standard deviation, ratio of the maximum to the minimum, maximum, arithmetic mean, and geometric mean among the term probabilities of all query terms. While the first baseline only uses the features extracted from query texts, the work of [17] additionally considers interaction behavior (e.g., clickthrough statistics) which can largely improve prediction performance. Thus, as Baseline 2, we implement the features proposed in [17], which showed best performance in that paper. For more robust baselines, we combine the features from the Baseline 1 and 2 to form Baseline 3. In addition, we develop Baseline 4 which uses all the attributes (see Section 4.1) as features but does not use the association rules. The purpose of Baseline 4 is to verify the effectiveness of our rule mining method (Section 4.2) by comparing a system that uses the association rules as features and another that uses the original binary attributes used to mine the rules.

5.2.3 Classification Results

To perform a fair comparison, we run the methods with various settings. First, to compare with Baseline 1, we exclude behavior attributes and generate association rules (Cat+Lex(NOBH)) because Baseline 1 uses only query texts for extracting its features and in our method only behavior attributes require out-of-query information (sequence of search actions). Second, to compare with Baseline 4, we use only categorical rules (Cat) because our objective is to verify the effectiveness of the rule mining method. Third, we combine our approach with the most robust baseline (Baseline 3) to verify further enhancements when using behavioral signals. Tables 6 and 7 (overleaf) show the classification results with SVM and LR, respectively. We measure Precision, Recall, and F1 for each class. The area under the receiver operating characteristic curve (AUC) is measured to identify overall classification performance. Since we focus on DSAT query identification, the metrics related to DSAT are more important.

First, as we intended, Baseline 3 performs better than Baseline 1 and 2. In both classifiers, Baseline 3 can significantly outperform the two baselines in terms of DSAT precision, AUC and all metrics regarding SAT. Second, the rule mining method (Section 4.2.1) is effective using both classifiers. Comparing Cat with Baseline 4, the performance of Cat is significantly better in most metrics (except DSAT Recall and SAT Precision using SVM). This is because a categorical rule itself is a meta-feature, one effective subset of the features in Baseline 4 (i.e., categorical attributes). In other words, learning with the rules, the classifier can find an effective linear combination of equally associated features (i.e., the rules). Third, concerning Cat+Lex (NOBH), our system can significantly outperform Baselines 1, 2, and 3 in every DSAT metric and AUC, which means that the rules are more effective than the existing predictors [10][17][25]. This result is important because as we described in Section 1, we attempt to identify dissatisfaction without behavior information (before searches terminate), and our system excluding behavior attributes is better than the baselines using the interaction features. Moreover, Cat+Lex using all proposed attributes can outperform the baselines as well. Fourth, we combine our system (Cat+Lex) with the robust baseline (Baseline 3), and in SVM experiments (Table 6), we observed significant improvements on overall classification performance (i.e., AUC). However, using LR (Table 7), the combination performs worse than our system; especially in DSAT Recall, where a significant falloff (about a 32% drop from the performance of Cat+Lex) is observed, but in AUC the performance slightly decreases and significance is not observed. This is because in LR experiments (Table 7), when compared to Cat+Lex, Baseline 3 performs poorly in terms of DSAT Recall (though in AUC, Baseline 3 is much better than Baseline 1 and 2) and combining with it is harmful to our system which basically performs much better than the baselines. On the other hand, in SVM experiments (Table 6), comparing to Cat+Lex, Baseline 3 is competitive and the combined approach can perform significantly better in AUC. Overall, our methods (Cat+Lex and Cat+Lex (NOBH)) are more effective at identifying DSAT than existing performance predictors [10][17][25] in two statistical learning frameworks.

5.3 Further Analysis

We provide a qualitative analysis of our approach via query examples. Table shows three sample rules generated by our system and example queries represented by the rules. In the first rule, the queries (instances) related to 0 (behavior category, which indicates more query and less search result exploration) and Society/ People are dissatisfaction. In particular, the users querying for [lisa hill] reformulated the query several times but only clicked on two results with short dwell time (less than 30 seconds).

Table 6: Classification results using SVM. Cat indicates using only categorical rules, Lex indicates using lexical rules, NOBH denotes excluding behavior attributes (Section 4.1.4), and Baseline4 uses categorical attributes as features (Section 5.2.2). In each column, a significant improvement over each baseline is marked by its number, e.g., B12 indicates improvement over Baseline 1&2. A † and ‡ denote a significant improvement over Cat and Cat+Lex, respectively. The paired t-test is performed with $p < 0.01$.

Method \ Metric	DSAT Precision	DSAT Recall	DSAT F1	SAT Precision	SAT Recall	SAT F1	AUC
Baseline 1 [10][25]	0.5951	0.6614	0.6265	0.6210	0.5277	0.5706	0.6427
Baseline 2 [17]	0.5965	0.6535	0.6237	0.6189	0.5393	0.5764	0.6425
Baseline 3 [10][17][25]	0.6932 B12	0.6742	0.6836 B12	0.7248 B12	0.7395 B124†‡	0.7321 B124†‡	0.7710 B124
Baseline 4	0.6911 B12	0.7297 B123	0.7099 B123	0.7165 B12	0.6685 B12	0.6917 B12	0.7557 B12
Cat	0.7119 B1234	0.7242 B123	0.7180 B1234	0.7201 B12	0.7052 B124	0.7126 B124	0.7794 B1234
vs. Baseline 4	*+3.01%*	*−0.75%*	*+1.14%*	*+0.50%*	*+5.49%*	*+3.02%*	*+3.14%*
Cat+Lex (NOBH)	0.7081 B1234	0.7348 B123	0.7212 B1234	0.7253 B12	0.6953 B124	0.7100 B124	0.7787 B1234
vs. Baseline 1	*+18.99%*	*+11.10%*	*+15.12%*	*+16.80%*	*+31.76%*	*+24.44%*	*+21.16%*
vs. Baseline 2	*+18.71%*	*+12.44%*	*+15.36%*	*+17.19%*	*+28.93%*	*+23.18%*	*+21.20%*
vs. Baseline 3	*+2.15%*	*+8.99%*	*+5.51%*	*+0.07%*	*−5.98%*	*−3.02%*	*+1.00%*
Cat+Lex	0.7167 B1234	0.7221 B123	0.7194 B1234	0.7210 B12	0.7125 B124	0.7167 B124	0.7831 B1234†
vs. Baseline 1	*+20.43%*	*+9.18%*	*+14.83%*	*+16.10%*	*+35.02%*	*+25.62%*	*+21.85%*
vs. Baseline 2	*+20.15%*	*+10.50%*	*+15.34%*	*+16.50%*	*+32.12%*	*+24.35%*	*+21.88%*
vs. Baseline 3	*+3.39%*	*+7.10%*	*+5.24%*	*−0.52%*	*−3.65%*	*+2.10%*	*+1.57%*
Cat+Lex+Baseline3	0.7124 B1234	0.7261 B123	0.7192 B1234	0.7534 B1234†‡	0.7360 B124†‡	0.7446 B1234†‡	0.8175 B1234†‡
vs. Baseline 3	*+2.77%*	*+7.70%*	*+5.21%*	*+3.95%*	*−0.47%*	*+1.71%*	*+6.03%*
vs. Cat+Lex	*−0.60%*	*+0.55%*	*−0.03%*	*+4.49%*	*+3.30%*	*+3.89%*	*+4.39%*

Table 7: Classification results using LR. The same notation is used as in Table 6. The paired t-test is performed with $p < 0.01$.

Method \ Metric	DSAT Precision	DSAT Recall	DSAT F1	SAT Precision	SAT Recall	SAT F1	AUC
Baseline 1 [10][25]	0.6386	0.5257 B3	0.5767	0.5414	0.6537	0.5923	0.6282
Baseline 2 [17]	0.6173	0.5718 B3	0.5937	0.5538	0.6146	0.5826	0.6338
Baseline 3 [10][17][25]	0.6936 B12	0.4778	0.5658	0.6429 B12	0.8160 B124†‡	0.7192 B124	0.7747 B124
Baseline 4	0.6789 B12	0.7319 B123	0.7044 B123	0.7092 B123	0.6536	0.6803	0.7529 B12
Cat	0.7155 B1234	0.7350 B123	0.7251 B1234	0.7276 B1234	0.7077 B124	0.7175 B124	0.7942 B1234
vs. B4	*+5.39%*	*+0.42%*	*+2.94%*	*+2.59%*	*+8.28%*	*+5.48%*	*+5.49%*
Cat+Lex (NOBH)	0.7128 B1234	0.7408 B1234	0.7265 B1234	0.7302 B1234	0.7015 B124	0.7156 B124	0.7938 B1234
vs. Baseline 1	*+11.62%*	*+40.92%*	*+25.99%*	*+34.87%*	*+7.31%*	*+20.82%*	*+26.36%*
vs. Baseline 2	*+15.47%*	*+29.56%*	*+22.38%*	*+31.85%*	*+14.14%*	*+22.82%*	*+25.24%*
vs. Baseline 3	*+2.77%*	*+55.04%*	*+28.40%*	*+13.58%*	*−14.03%*	*−0.50%*	*+2.47%*
Cat+Lex	0.7163 B1234	0.7363 B123	0.7262 B1234	0.7287 B1234	0.7084 B124	0.7184 B124	0.7953 B1234
vs. Baseline 1	*+12.17%*	*+40.06%*	*+25.92%*	*+34.60%*	*+8.37%*	*+21.30%*	*+26.60%*
vs. Baseline 2	*+16.04%*	*+28.77%*	*+28.77%*	*+31.58%*	*+15.26%*	*+23.31%*	*+25.48%*
vs. Baseline 3	*+3.27%*	*+54.10%*	*+28.34%*	*+13.35%*	*−13.19%*	*−0.11%*	*+2.66%*
Cat+Lex+Baseline 3	0.7148 B1234	0.5036 B3	0.5909 B123	0.6504 B123	0.8207 B124†‡	0.7257 B1234†	0.7882 B1234
vs. Baseline 3	*+3.06%*	*+5.40%*	*+4.43%*	*+1.17%*	*+0.58%*	*+0.91%*	*+1.74%*
vs. Cat+Lex	*−0.21%*	*−31.60%*	*−18.63%*	*−10.75%*	*+15.85%*	*+1.01%*	*−0.89%*

Table 8: Sample rules and query examples. BH denotes Behavior attributes (Section 4.1.4), ODP denotes ODP attributes (Section 4.1.1), and WIKI denotes Wikipedia attributes (Section 4.1.3). For each rule, the consequent ({DSAT}) is omitted.

No.	Rule	Example Query	Label
1	{0[BH], Society/People[ODP]}	lisa hill	DSAT
2	{automobiles[WIKI], "warranty"}	1989 toyota mr2 warranty	DSAT
		2011 suzuki sx4 factory warranty	SAT
3	{motor vehicle company of Italy[WIKI], ¬aventador, ¬fiat, "ferrari"}	used ferrari cars scottsdale	DSAT
		certified ferrari cars phoenix	DSAT

The second rule describes a set of underperforming queries for auto warranty information. We found that sometimes the warranty information of older models (e.g., [1989 toyota mr2]) is not readily located whereas the information for newer models (e.g., [2011 suzuki sx4]) is easily found. The third rule contains negation, and shows that some users are dissatisfied with results for a group of queries related to "ferrari" and not containing "fiat" or "aventador".

6. CONCLUSION

In this paper, we proposed a framework to automatically generate association rules to identify underperforming queries. In order to build effective rules, we first generate topical attributes recognized from query text and formulate association rules that discover frequent patterns of the attributes for identifying dissatisfaction queries. Then, we apply a decision tree learning to identify discriminative keywords of dissatisfaction queries and combine the keywords with the association rules. In experiments, we verified the effectiveness of our system in the task of dissatisfaction query classification in comparison to existing query performance prediction baselines. Considering previous work on modeling user satisfaction, the advantage of our method is identifying dissatisfaction at query time and our system can provide evidence of dissatisfaction to search engines before users abandon searches. For future work, we plan to devise effective methods to improve the instances of search dissatisfaction identified by our method.

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. *SIGIR*, 345–354.

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

[3] Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. *VLDB*, 487–499.

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

[5] Andrew, G. and Jianfeng, G. (2007). Scalable training of L_1-regularized log-linear models. *ICML*, 33–40.

[6] Aula, A., Khan, R.M., and Guan, Z. (2010). How does search behavior change as search becomes more difficult? *CHI*, 35–44.

[7] Bennett, P.N., Svore, K., and Dumais, S.T. (2010). Classification-enhanced ranking. *WWW*, 111–120.

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

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

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

[11] Coenen, F., Goulbourne, G., and Leng, P. (2004). Tree structures for mining association rules. *Data Mining and Knowledge Discovery*, 8(1): 25–51.

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

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

[14] Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

[15] Finkel, J., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. *ACL*, 363–370.

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

[17] Guo, Q., White, R.W., Dumais, S.T., Wang, J., and Anderson, B. (2010). Predicting query performance using query, result, and user interaction features. *RIAO*, 198–201.

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

[19] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I.H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1): 10–18.

[20] Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. *SIGMOD*, 1–12.

[21] Hassan, A. (2012). A semi-supervised approach to modeling web search satisfaction. *SIGIR*, 275-284.

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

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

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

[25] He, B. and Ounis, I. (2006). Query performance prediction. *Information System*, Vol. 31(7), 585–594.

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

[27] Jones, R. and Klinkner, K. (2008). Beyond the session timeout: automatic hierarchical segmenting of search topics in query logs. *CIKM*, 699–708.

[28] Leskovec, J., Dumais, S., and Horvitz, E. (2007). Web projections: learning from contextual sub graphs of the web. *WWW*, 471–480.

[29] Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: primal estimated sub-gradient solver for SVM. *ICML*, 807–814.

[30] Wang, C., Hsu, B., Chang, M., and Kiciman, E. (2012). Simple and knowledge-intensive generative model for named entity recognition. *Microsoft Research Technical Report*.

[31] Wang, K., Thrasher, C., Viegas, E., Li, X. and Hsu, P. (2010). An overview of Microsoft web n-gram corpus and applications. *NAACL HLT Demo Session*, 45–48.

[32] Weiss, Y., Torralba, A., and Fergus, R. (2008). Spectral hashing. *NIPS*, 1753–1760.

[33] Zhao, Y., Scholer, F., and Tsegay, Y. (2008). Effective pre-retrieval query performance prediction using similarity and variability evidence. *ECIR*, 52–64.

[34] Zhou, Y., and Croft, W.B. (2006). Ranking robustness: a novel framework to predict query performance. *CIKM*, 567–574.

[35] Zhou, Y. and Croft, W.B. (2007). Query performance prediction in web search environments. *SIGIR*, 543–550.