Mining Broad Latent Query Aspects from Search Sessions

Xuanhui Wang*
Dept. of Computer Science
University of Illinois at
Urbana-Champaign
Urbana, IL 61801
xwang20@cs.uiuc.edu

Deepayan Chakrabarti
Yahoo! Research
701 1st Avenue
Sunnyvale, CA 94089
deepay@yahoo-inc.com

Kunal Punera
Yahoo! Research
701 1st Avenue
Sunnyvale, CA 94089
kpunera@yahoo-inc.com

ABSTRACT

Search queries are typically very short, which means they are often underspecified or have senses that the user did not think of. A broad latent query aspect is a set of keywords that succinctly represents one particular sense, or one particular information need, that can aid users in reformulating such queries. We extract such broad latent aspects from query reformulations found in historical search session logs. We propose a framework under which the problem of extracting such broad latent aspects reduces to that of optimizing a formal objective function under constraints on the total number of aspects the system can store, and the number of aspects that can be shown in response to any given query. We present algorithms to find a good set of aspects, and also to pick the best k aspects matching any query. Empirical results on real-world search engine logs show significant gains over a strong baseline that uses single-keyword reformulations: a gain of 14% and 23% in terms of human-judged accuracy and click-through data respectively, and around 20% in terms of consistency among aspects predicted for "similar" queries. This demonstrates both the importance of broad query aspects, and the efficacy of our algorithms for extracting them.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Search process, Query formulation

General Terms: Algorithms

Keywords: Latent user intents, query aspects, search sessions.

1. INTRODUCTION

Search engines have become the primary mode of discovering and accessing content for a large fraction of web users. However, even though they use search engines for critical information access tasks, users are remarkably laconic in describing their information needs [15, 19], resulting in vague queries. This is due to several reasons. (1) Users often use search engines for performing research on unfamiliar topics. They might skip important terms in search queries simply because they are unacquainted with the topic-specific vocabulary. (2) They may be aware of the terms but believe them to be redundant; they may be unaware of the multiple ambiguous senses of their incomplete queries. (3) Search engines themselves reinforce this behavior by not properly taking into account the extra information when the users do provide long descriptive queries.

Discovering the intent latent in vague user queries is an active area of research. Methods proposed in prior art can mainly be grouped into two paradigms: *query expansion* and *query suggestions*. Query expansion approaches implicitly add extra terms to the search query [3, 12]. Such terms could include synonyms of query words, stemmed words, corrections of spelling mistakes, and so on. A special class of methods employ Pseudo-Relevance Feedback approaches [5, 22] to expand queries under the assumption that the top-k documents retrieved for the original query are relevant. In many cases, however, queries have several possible intents. For example, the user intent behind a query like "Canon EOS-40D Digital SLR" could be to find reviews and ratings for the camera, or shopping sites to buy the camera, or fan forums that discuss the camera, or simply to find information about the camera. As there is no single dominant sense of the query, no implicit query expansion can capture the true user intent with any confidence. In these cases, the best course of action is to help users explicitly specify the exact sense they had intended for their query; query expansion methods are not applicable to this problem.

The second type of methods, query suggestions, encourage users to explicitly specify the hidden intent behind their queries by offering them a list of semantically related queries. Users can then pick one of them to further refine their query. Examples include the "Also try" suggestions in Yahoo! and "Related searches" in Live Search. Yahoo! also provides a search interface called "Search Assist" that lists concepts related to the user's query, which the user can then use to restrict her search. A similar "Refinement" functionality for queries in specific domains like Health is provided by the Google search engine. In addition to these examples, several methods have been proposed in academic venues to find semantically related queries [8, 11, 17, 21]. A universal characteristic, however, of these methods is that the suggestions they offer are narrow and specific to the original query. As we explain next, this is not useful for finding broad aspects as is the goal of this paper.

Broad Latent Aspects of Search Queries. In our analysis of the Yahoo! search query logs we noticed the existence of many latent intents that are very broad, and applicable to many classes of queries. We call these intents *Broad Latent Aspects* of queries. An example of this is "Reviews and Ratings." Users often query for product names but neglect to mention the term "review," even when what they really want are reviews for the product. The "Reviews and Ratings" aspect is broad as well, since it is a latent aspect in queries for products, movies, restaurants, and many others. Other examples include "Pictures," when the query mentions a well-known landmark or person, and "Download," when the query only names a software title or a famous photograph or painting, and so on.

KDD'09, June 28–July 1, 2009, Paris, France.

* This work was done when the author was an intern with Yahoo! Research. The author is supported at UIUC by a Yahoo! Ph.D. fellowship.

Broad latent aspects of queries are characterized by the following two properties. First, these intents are applicable to a broad set of queries. As mentioned above, the "Reviews and Ratings" aspect is useful for many types of product and service queries. Second, users frequently do not specify terms indicating such aspects in their queries. For instance, while many users do specify the term "review" in their queries, a significant fraction only query for the product or service name even when they explicitly intended to find reviews. Together, these imply that a small set of such aspects, once discovered, can be used in inferring user latent intent for many incomplete queries. Since these aspects are *global* and reflect users' major latent intents, they can potentially be used to design specialized search services such as vertical searches.

In this paper, we propose methods to mine a list of broad latent aspects of search queries from query logs and present an approach to decide which broad aspects apply to future, possibly unseen, search queries. These query aspects are used to aid users in clarifying their information needs and providing the best search results.

Query Reformulations in Search Sessions. We extract broad latent aspects in queries by mining search engine user sessions. In particular, we model the query reformulations within sessions to discover these aspects.

User search sessions contain a lot of semantic information. As an example of a query reformulation in a user session, consider a user who wants to read reviews for the "Canon EOS-40D Digital SLR" but only provides the model name as the query. When the search engine responds with a results page full of links to where the camera can be bought, the user reformulates the query by inserting the term "reviews" at the end. The user then finds the desired result on the subsequent search results page and clicks on it. For the purposes of our work, "Canon EOS-40D Digital SLR" is referred to as the *original query* and "reviews" as the *query qualifier*. Such a reformulation can be used to discover the user's latent intent. We thus extract such instances of query reformulations from user sessions, and aggregate the query qualifiers to construct broad query aspects.

Broad Latent Aspects vs. Query Suggestions. The extraction of broad aspects from query logs, and their use in query refinement, has several advantages over standard query suggestion methods presented in the literature. The first advantage has to do with the discovery and use of broad aspects and query suggestions. The broad nature of the query aspects ensures that enough data is available to reliably construct these aspects and predict when they apply to user queries. This is in contrast to query suggestions that are often applicable to specific queries and hence learned from significantly lesser amount of data. The availability of more data for analysis also implies that we can avoid presenting the user with redundant query refinement options, as is often the case with query suggestions. Finally, since by definition there are fewer broad aspects of queries than query suggestions, we can learn more expensive model for them and also maintain them better (the maintenance at commercial search engines might involve costs of manual effort).

The second, more central, advantage is subtle, and concerns the way users navigate the search results page. It has been shown in user eye-tracking studies[1] as well as by modeling user clicking behavior [10] that users scan search result pages extremely fast and do not make complete determination of the relevance of results before clicking. Users get used to repetitive features in the search results page and use them to make clicking decisions. For example, the presence of bold words in the title of the result indicates to users that the title matched the query very closely while the indented search result indicates to the user that this search result is somehow related to the preceding one. When users are exposed to query suggestions, which by definition are specialized to the current query, they have to carefully read each suggested queries in order to decide whether to click on them. Since the users scan result pages very fast, they often skip the suggested queries as irrelevant content. By using a limited number of broad query aspects as options for refinement we seek to get the user accustomed to them. The users will then need less attention to interpret the aspects; indeed, they may even come to expect the aspects from the search engine – e.g., the "Reviews and Ratings" aspect, when they search for a product name.

[1]such as `www.checkit.nl/pdf/eyetracking_research.pdf`

Our Contributions. Our primary contributions are fourfold. First, we define broad query aspects and provide a clear problem formulation based on optimizing a well-defined objective function. The formulation directly models the search activity of users, specifically, the query reformulations. To the best of our knowledge, ours is the first attempt to use query reformulations for this purpose. Second, we propose algorithms that extract these broad query aspects by mining query reformulations in user session logs. Our proposed algorithms seek to generate semantically coherent aspects which can optimally cover the reformulations of the original queries. Third, we provide an optimal algorithm to pick the k most relevant aspects for any query, given a predefined set of latent query aspects generated by any algorithm. This algorithm might be of independent interest as well. Finally, we empirically demonstrate the accuracy of our proposed method on a large real-world corpus of search logs: a gain of 14% and 23% in terms of human-judged accuracy and click-through data respectively, and around 20% in terms of consistency among aspects predicted for "similar" queries.

2. RELATED WORK

Aiding web search users in finding relevant web pages is an active area of research. Prior work in this field can broadly be categorized into the following groups.

Query Suggestions. There is considerable literature on automatically suggesting queries that are related to the user's information need. Jones et al. [17] learn a supervised model to select a suggestion for a query from a list of candidates by using features that depend on the query and the candidate. Cucerzan and Brill [11] present an approach that ranks candidates by the conditional probability of seeing them in the same session as the current user query. Vlachos et al. [21] and Chien and Immorlica [8] operate on the assumption that semantically related queries have similar temporal behavior of query occurrences. There are also many query expansion techniques that use pseudo-relevance feedback [5, 22] and click-through behavior [3, 12].

Query Suggestions via Context. There is some work on personalizing the query suggestions to the searching history of the user. Cao et al. [6] look at the past sequence of queries issued by the user while making a query recommendation. In a similar work Boldi et al. [4] solve an ATSP problem to obtain the sessionization of user history and then use queries from the same session to bias query suggestions. Chirita et al. [9] expand a user's query with terms collected in user's profile to bias search results.

Concept Based Query Suggestions. The above methods all find queries related to the current user information need, and this can be error prone if the information need is rare or novel. A possible solution is to aggregate queries into concepts which are then used to map related queries to the user's current information need. Fonseca et al. [13] use association rule mining to fetch past queries related to the current query, which are then clustered via locating cycles in a specially constructed query-relations graph. These clusters are then

shown to the user, who picks the cluster that is to be used to expand the current query. Cao et al. [6] recommend queries by first clustering them into concepts in an offline process. There has also been significant work on assigning labels to query clusters; for example, Pasca and Van Durme [18] use correlations in query logs and web documents to create and label query clusters. Finally, Fuxman et al. [14] use a pre-defined taxonomy of concepts to learn a mapping from query to concept. This mapping is then used to recommend related queries in the context of sponsored search.

Differences in Our Work. As mentioned in Section 1, we discover broad latent query aspects from reformulation activity in user sessions. Unlike query expansion [3, 12], these query aspects are designed for situations when there is no one dominant latent intent to a query. In contrast to query suggestions [8, 11, 17, 21], query aspects are *global*, applicable to many classes of queries, and can also be used to design specialized search services such as vertical searches. These aspects must in some sense be orthogonal to the topical nature of queries. Moreover, in our approach we seek to find aspects that users often neglect to include in their queries. We accomplish this by explicitly modeling the query reformulations in search sessions. This is different from all the works mentioned above.

3. OUR APPROACH

In the Introduction, we motivated the use of query reformulations in search sessions for extracting broad latent query aspects. In this section, we first present a concrete formulation of our problem. The end result is an encoding of our problem in a precise objective function. We then give algorithms that directly optimize this objective function to discover broad latent aspects and to pick appropriate aspects in response to future queries.

3.1 Problem formulation

Let A denote a set of broad latent query aspects. Each query aspect a_i is in turn a set of *query qualifiers* – terms that are added to an original query during reformulations. Note that A does not have to cover the set of all query qualifiers exhaustively or mutually exclusively; A need not be a partitioning. However, primarily for the purposes of efficiency as shown later, we assume in our work that query aspects are disjoint.

The problem setting can now be stated compactly as follows. Our system discovers and maintains a set A of N query aspects. On receiving any original query q at run-time, it select k out of the N aspects and presents them to the user along with search results for q. We want to (a) "cover" the set $\ell(q)$ of expected qualifiers of q as precisely as possible; the k query aspects must include most of the qualifiers in $\ell(q)$ but also exclude most extraneous qualifiers, and (b) "cover" as many queries as possible from the search engine workload.

The restriction on N query aspects is due to costs associated with discovering and maintaining query aspects in a large search system, while the restriction on k query aspects derives from constraints on the real-estate on the search results page as well as user's attention. Its easy to see that these restrictions result in a trade-off between the two requirements of our approach: to cover as many as queries as possible, as precisely as possible.

This trade-off needs to be encapsulated in a single reward function, which will then let us compare different solutions (each solution being a set A of N query aspects). Following the literature on information retrieval, natural choices for such a reward function are the *precision* and *recall* of the solution. The precision of a query aspect a_i is the fraction of its qualifiers that it shares with $\ell(q)$: $\text{prec} = |a_i \cap \ell(q)|/|a_i|$. The recall is the fraction of qualifiers in $\ell(q)$ that are present in a_i: $\text{rec} = |a_i \cap \ell(q)|/|\ell(q)|$. Precision rewards query aspects with few extraneous qualifiers, while recall rewards aspects which cover most qualifiers of the original query. Together, these are exactly our two desiderata. A standard way to combine them is via the F-measure, which is the harmonic mean of precision and recall [2]:

$$\text{F-measure} = \frac{2}{\frac{1}{\text{prec}} + \frac{1}{\text{rec}}} = 2 \cdot \frac{\text{prec} \cdot \text{rec}}{\text{prec} + \text{rec}}. \quad (1)$$

The F-measure ranges from 0 to 1, with 0 indicating no similarity and 1 representing identical sets.

Now, we can define the goodness of a solution A as the F-measure between the set $\ell(q)$ of qualifiers of a query q and the union of the k query aspects picked for it, summed over all queries in the query workload Q and weighted by their frequencies $n(q)$:

$$R(Q, A) = \sum_{q \in Q} n(q) \cdot \max_{a_1, a_2, \ldots, a_k \in A} F\left(\ell(q), \cup_{i=1}^{k} a_i\right) \quad (2)$$

This reward function represents a trade-off between two advantageous properties: it encourages "covering" as many qualifiers of $\ell(q)$ as possible, while penalizing for both qualifiers that are not covered, and overly broad query aspects. Note that the latter is a problem with objective functions based on several other common similarity measures: e.g., neither precision-at-k nor cosine similarity penalize for uncovered qualifiers. The two properties cannot be simultaneously satisfied to the maximum possible extent because of constraints on both the number of total aspects N and the number of aspects k that can be used to "cover" any one query, making the trade-off necessary.

While the standard F-measure indeed has several desirable properties, it does not take into account the relative frequencies of elements in the sets being compared. This is important in our setting: the query aspects presented in response to a query should preferably be biased towards the most frequent reformulations of that query. For example, suppose an original query, say *"mariah carey"*, has three qualifiers: *pictures* (weight 100), *wallpaper* (weight 10), and *photoshoot* (weight 5). This tells us that a user querying for *"mariah carey"* is 10 times more likely to be looking for simple pictures than for desktop wallpapers. Then a query aspect consisting of *pictures*, *wallpaper*, and *photoshoot*, each weighted equally, does not provide a user with an appropriate choice for reformulation. A user interested in, say, *pictures* will only have a 1/3 chance of finding this aspect relevant. Ideally, the qualifiers in an aspect should be weighted with exactly the same relative weights, i.e., $100 : 10 : 5$. However, the F-measure in Equation 1 will score both the equal-weight and biased-weight versions of this aspect identically, considering both to perfectly cover the original query. Hence, it is crucial for us to use a similarity function that properly handles weights, and this is what we discuss next.

Weighted F-measure. We must first extend the notation to the case of weights. Slightly abusing notation, we will use $\ell(q)$ to represent not only the set of qualifiers of query q but also the *vector* of weights of those qualifiers. Weights are non-zero only for qualifiers that appear in the historical data with q. Similarly, let $a_i \in A$ now represent the weight vector for a query aspect. We desire a similarity function between $\ell(q)$ and a_i when both are weighted.

To achieve this, the trade-off represented by Equation 1 must now be recast in terms of weights. We accomplish this by positing that a suitable weighted similarity measure should possess the following two characteristics: **(P1)** An a_i that has high weights exactly on the terms on which $\ell(q)$ also puts high weights should achieve a high similarity score, and **(P2)** if we take a qualifier that exists in $\ell(q)$ but not in a_i, and add it to a_i with a very small weight ϵ, the similarity

should necessarily increase. Of course, if this new addition were to be weighted too heavily in a_i, then the similarity score may either increase or decrease, depending on its relative weight in $\ell(q)$.

Keeping these properties in mind, we propose extending the F-measure to the weighted case when a_i and $\ell(q)$ are vectors.

$$\begin{aligned} \text{prec} &= \frac{a_i \cdot \ell(q)}{a_i \cdot a_i} \\ \text{rec} &= \frac{a_i \cdot \ell(q)}{\ell(q) \cdot \ell(q)} \\ F\left(\ell(q), a_i\right) &= 2 \cdot \frac{\text{prec} \cdot \text{rec}}{\text{prec} + \text{rec}} = 2 \cdot \frac{a_i \cdot \ell(q)}{\|a_i\|^2 + \|\ell(q)\|^2} \quad (3) \end{aligned}$$

Thus, the set size computations of Equation 1 are replaced by dot products between vectors.

It is clear from Equation 3 that $F(.,.)$ attains high values when items in both sets have *equal* weights, and low values when the weights diverge, thus satisfying property **(P1)**. It can be shown that **(P2)** holds as well, as long as a_i is non-empty. The only remaining question is the setting of weights for the qualifiers in a_i and $\ell(q)$.

Weighing qualifiers in $\ell(q)$ and a_i. The weights on the two sets serve different purposes and this dictates useful ways of setting them. Our goal is to ensure that for a user issuing query q, the system suggests aspects that match what we expect to be reformulations for q. Hence, a natural value for the weight of a qualifier m in $\ell(q)$ is the number of times we expect to see m in a reformulation of q (we can estimate this from historical data); we call this the *conditional* frequency. On the other hand, when the system suggests an aspect a_i in response to a query q, the user's perceived relevance of a_i to q depends on the qualifiers that together comprise a_i. The weight of a qualifier l in a_i should reflect the degree to which it dominates other qualifiers in a_i. We estimate this weight as the global frequency of seeing l as a qualifier in any reformulations over the entire historical data; we call this the *global* frequency of l.

In this weighting scheme, for the F-measure to make sense, the weights on members of a_i and $\ell(q)$ must be on the same scale. This is unlikely to be the case since global frequencies will tend to be much larger than conditional frequencies. An obvious way to ensure similar scales is to normalize the weights in a set to sum to 1. This fix, however, runs afoul of property **(P2)** mentioned above. To illustrate this, let us consider the previous example – the query *"mariah carey"*, with set $\ell(q)$ of qualifiers being *pictures* (unnormalized weight 100), *wallpaper* (unnormalized weight 10), and *photoshoot* (unnormalized weight 5). Then a query aspect a_i consisting of *pictures* (unnormalized weight 10) and *wallpaper* (unnormalized weight 1) would score well since the highly weighted qualifiers of the original query are also highly weighted in the query aspect. However, if we now add *photoshoot* to our query aspect a_i, then the normalization would *reduce* the relative weights for *pictures* and *wallpaper*, and by implication their contributions to the similarity score. This could lead to an overall decrease in the similarity between $\ell(q)$ and the new a_i.

To bring the weights to the same scale while satisfying both **(P1)** and **(P2)**, we perform the following normalization: For each q, we scale all weights in $\ell(q)$ by a factor specific to q such that, after normalization, the squared 2-norm of $\ell(q)$ (i.e., $\|\ell(q)\|^2$) equals the sum of squares of the global frequencies of the members of the set $\ell(q)$. For example, if $\ell(q)$ has no overlaps with the qualifiers of any other query, then no scaling would be performed, which is intuitive since the conditional frequencies in this case *are* the global frequencies. Also, if there is a query aspect a_i with exactly the same terms as $\ell(q)$, then it would have the same frequencies as $\ell(q)$ as well, so the vectors a_i and $\ell(q)$ would be identical and the similarity measure $F\left(\ell(q), a_i\right)$ would achieve its highest possible value of 1. It may easily be seen that with this normalization scheme, $F(.,.)$ satisfies both desired properties.

Measure	α_X	β_X	$f_X(Y)$	$g_X(Y)$
F (Eq 5)	0	$\|X\|^2$	$\|X \cdot Y\|$	$\|Y\|^2$
Jaccard [20]	0	$\|X\|$	$\|X \cap Y\|$	$\|Y \setminus X\|$
Ext. Jaccard [7]	0	$\sum_j X_j$	$\sum_j \min(X_j, Y_j)$	$\sum_j \max(Y_j - X_j, 0)$

Table 1: Similarity measures as special cases of Equation 6

Finally, we can formally state our problem in its entirety:

Problem 1: *Find a set A of N query aspects so as to maximize*

$$R(Q, A) = \sum_{q \in Q} n(q) \cdot \max_{a_1, a_2, \ldots, a_k \in A} F\left(\ell(q), \cup_{i=1}^{k} a_i\right) \quad (4)$$

where,

$$F\left(\ell(q), \cup_{i=1}^{k} a_i\right) = 2 \cdot \frac{\textit{prec} \cdot \textit{rec}}{\textit{prec} + \textit{rec}} = 2 \cdot \frac{\sum_{i=1}^{k} a_i \cdot \ell(q)}{\sum_{i=1}^{k} \|a_i\|^2 + \|\ell(q)\|^2} \quad (5)$$

$$\|\ell(q)\|^2 = \sum_{m \in \ell(q)} \textit{global-freq}(m)^2 \quad \forall q \in Q$$

The above problem formulation essentially involves two problems: (1) find the best set of query aspects A, and (2) for any given query, pick the best k query aspects from the set A. Since the solution to the first problem depends on our ability to solve the second, we present, in Section 3.2, an algorithm for the second problem and prove its optimality. This algorithm, combined with a clustering technique, then leads to our proposed solution for the first problem, which is discussed in Section 3.3.

3.2 Picking the best k aspects

Given a set A of query aspects, and a query q, how should we pick k aspects $a_1, \ldots, a_k \in A$ so as to maximize the similarity measure $F\left(\ell(q), \cup_{i=1}^{k} a_i\right)$? We present a general solution that can maximize any similarity function of the form

$$h_X(Y) = \frac{\alpha_X + \sum_i f_X(y_i)}{\beta_X + \sum_i g_X(y_i)}, \quad (6)$$

where $Y = \{y_1, \ldots, y_k\}$ (k fixed) is a set that must be picked from a universe $\mathcal{Y}$ of possible items, α_X and β_X are constants with $\beta_X > 0$, the function $g_X(.)$ is non-negative, and all functions and constants are indexed by an X that represents any known and relevant data. The connection to our problem of picking query aspects is clear: The universe $\mathcal{Y}$ corresponds to A, with each y_i corresponding to some weighted query aspect. Similarly, X corresponds to the weighted $\ell(q)$, which is known. When all the aspects are mutually exclusive, our $F(.,.)$ objective function falls under this framework, along with several other similarity functions (Table 1). Thus, in addition to being applicable to our particular problem, the solution to Equation 6 might be of independent interest as well.

The difficulty in solving this problem stems primarily from an important, and somewhat counter-intuitive, consequence of Equation 6: The solution for large k need not be a superset of the solution for small k. Consider the following example. Let $\alpha_X = 0$, and $\beta_X = 10$. Let $\mathcal{Y} = \{y_1, y_2, y_3\}$ with the $f_X(y_i) = (1, 1, 2)$ and $g_X(y_i) = (1, 1, 10)$ for $i = \{1, 2, 3\}$. Now, if $k = 1$, the optimal $Y^* = \{y_3\}$, since this yields the objective value $h_X(\{y_3\}) = (0 + 2)/(10 + 10) = 1/10$ while $h_X(\{y_1\}) = h_X(\{y_2\}) = (0 + 1)/(10 + 1) = 1/11 < h_X(\{y_3\})$. However, if $k = 2$, the optimal $Y^* = \{y_1, y_2\}$, since $h_X(Y^*) = (0 + 1 + 1)/(10 + 1 + 1) = 1/6$, while the other solutions give smaller objective values: $h_X(\{y_1, y_3\}) = h_X(\{y_2, y_3\}) = (0+1+2)/(10+1+10) = 1/7$.

Algorithm 1 Pick-k

1: **input:** set size k, universe of items $\mathcal{Y}$, α, β, $f(.)$, $g(.)$
2: $Y \leftarrow \{\phi\}$, $n \leftarrow k$, $\alpha' \leftarrow \alpha$, $\beta' \leftarrow \beta$
3: **while** $n > 0$ **do**
4: $M \leftarrow \left\{ \arg\max_s \frac{\alpha'/n + f(s)}{\beta'/n + g(s)} \right\} \qquad (s \in \mathcal{Y} \setminus Y)$
5: If $|M| > n$, then keep any n elements in M and throw away the rest
6: $Y \leftarrow Y \cup M$
7: $\alpha' \leftarrow \alpha' + \sum_{m \in M} f(m)$
8: $\beta' \leftarrow \beta' + \sum_{m \in M} g(m)$
9: $n \leftarrow n - |M|$
10: **end while**
11: **output:** picked elements $Y \subseteq \mathcal{Y}$

This shows, among other things, that dynamic programming methods would not work for this problem. The set cover problem has a similar flavor, but it requires all elements in the given set to be covered and is thus NP-Complete, whereas our formulation allows for some elements to be left uncovered. This fact, along with the special structure of Equation 6, allows us to *optimally* solve our problem in polynomial time.

Algorithm Description. Algorithm 1 presents the Pick-k algorithm for picking the set of elements Y from $\mathcal{Y}$. Since X is known in any instance of the problem, we drop the subscript from all terms in the interests of clarity. Starting from an empty set $Y = \{\phi\}$, our proposed algorithm proceeds step by step, adding items from $\mathcal{Y}$ to Y in a greedy manner. However, the function that is maximized in each iteration keeps changing. The heart of the algorithm is in step 4, where the best items in each iteration are picked according to a function that *depends on the number of elements n yet to be picked*. Typically, there will only be one best element in M, though the general version presented in Algorithm 1 can handle ties as well. The time complexity of the algorithm is $O(k|\mathcal{Y}|)$ since there are k iterations, and all $|\mathcal{Y}|$ items must be considered in step-4 of each iteration. For Problem 1, this becomes $O(kN)$, where N is the number of query aspects.

Proof of Optimality. We need to prove that Algorithm 1 picks the optimal $Y^* = \arg\max_Y h(Y)$. Here, we provide a proof sketch, with the full proof being deferred to the appendix due to lack of space[2].

For ease of exposition, assume that there are no ties in step 4 and each iteration adds only one element to Y (the proof can be easily extended to cover that case of multiple additions per iteration).

The algorithm maximizes Equation 6 by solving a sequence of sub-problems. Suppose that the set $Y^*_{(k-n)} = \{y^*_1, \ldots, y^*_{k-n}\}$ is known to be a subset of Y^* of size $k - n$. Now, for any set $W \subseteq \mathcal{Y} \setminus Y^*_{(k-n)}$ such that W has exactly n elements, we use Equation 6 to get:

$$h(Y^*_{(k-n)} \cup W) = \frac{\alpha + \sum_{y \in Y^*_{(k-n)}} f(y) + \sum_{w \in W} f(w)}{\beta + \sum_{y \in Y^*_{(k-n)}} g(y) + \sum_{w \in W} g(w)} \quad (7)$$

Let $W^* = \arg\max_W h(Y^*_{(k-n)} \cup W)$. Now, we can define the following sub-problem:

Problem 1': *Given the 4-tuple $(Y^*_{(k-n)}, \alpha, \beta, n)$, find any one element $w^* \in W^*$.*

Next, we relate the solution of the sub-problem to the optimal solution Y^*.

LEMMA 1. *$w^* \in W^* \Rightarrow w^* \in Y^*$*

[2]The appendix has been submitted as supplementary material, and is also available online at `http://www.cs.cmu.edu/~deepay/AspectsAppendix.pdf`

LEMMA 2. *In Problem 1', the 4-tuple $(Y^*_{(k-n)}, \alpha, \beta, n)$ and the 4-tuple $\left(\{\phi\}, \alpha + \sum_{y \in Y^*_{(k-n)}} f(y), \beta + \sum_{y \in Y^*_{(k-n)}} g(y), n\right)$ are equivalent.*

The optimality of Algorithm 1 can now be proved in two stages. First, assuming the correctness of our solution to each sub-problem, we show that the *sequence* of sub-problems generated by steps 6-9 of our algorithm is correct. Second, we show that each sub-problem is solved correctly (step 4). In the following, α' and β' refer to the variables updated in steps 7-8 of Algorithm 1.

THEOREM 1. *Assuming that step 4 of Algorithm 1 correctly solves the sub-problem $(\{\phi\}, \alpha', \beta', n)$, the algorithm returns the optimal result Y^*.*

PROOF SKETCH. The proof is by induction on the number of iterations. Each iteration solves a particular sub-problem $(\{\phi\}, \alpha', \beta', n)$, which is equivalent to (Y, α, β, n) by Lemma 2. Here $Y = Y^*_{(k-n)}$ is the current solution of step 6. Then, by Lemma 1, each iteration yields a new element of Y^*. The full proof is in the appendix. □

THEOREM 2. *Step 4 of Algorithm 1 correctly solves the sub-problem $(\{\phi\}, \alpha', \beta', n)$, i.e., the element s^* picked in step 4 of an iteration belongs to the optimal solution: $s^* \in W^*$.*

PROOF SKETCH. We prove that if s^* does *not* belong to W^*, then we can construct a new set Z which replaces an element of W^* by s^* such that $h(Z) > h(W^*)$, leading to a contradiction. The full proof is provided in the appendix. □

3.3 Generating the set A of query aspects

In the previous section, we proposed a method to pick the best k query aspects for any given query. Now, we will present our approach for constructing the set A of N query aspects.

Construction of the set A must adhere to the central goals and constraints of the problem formulated in Section 3.1. Recall that only N total aspects can be stored by the system, and only k aspects can be shown in response to a query. This leads us to construct a set A in which query qualifiers are grouped together into broad aspects so that only a few aspects can be used to cover a significant fraction of the query workload. On the other hand, the goal of providing precise aspects to users prompts us to keep semantically distinct query qualifiers in separate groups. This trade-off between coverage and specificity of aspects leads us to propose a solution based on clustering of query qualifiers.

Clustering query qualifiers. In our approach, each query aspect is modeled as a cluster of "similar" query qualifiers. Two qualifiers are considered similar if the corresponding sets of original queries to which they tend to be added are similar. While any clustering algorithm and similarity metric can be used within our framework, in this paper we extend the well-known Star Clustering algorithm [1] for this purpose. We chose Star Clustering as it has been shown to be effective over many different datasets [1], and has several characteristics that help it optimize our objective function of Equation 4, albeit indirectly. Next we describe the extended algorithm, and then discuss the reasons behind its success on our problem.

Algorithm 2 presents our extended version of the star clustering algorithm. Each qualifier v in the historical data is attached to a vector, whose dimensions are the set of original queries Q, and whose values are the frequencies with which v occurs as a qualifier for each $q \in Q$. Pairwise cosine similarities are then computed between qualifiers (typically only the most frequent $10,000$ qualifiers in order to reduce computation), and only those pairs with similarity above a given threshold σ are retained. This creates a graph among

Algorithm 2 Modified Star Clustering

input: set of qualifiers $\mathcal{V} = \cup_{q \in Q} \ell(q)$, qualifier frequencies $L(v) \forall v \in \mathcal{V}$, threshold σ, N
Create a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of qualifiers, and $\mathcal{E} = \{(i,j) | \text{cosSim}(i,j) > \sigma\}$
$n \leftarrow 0$, Left $\leftarrow \mathcal{V}$, $A \leftarrow \{\phi\}$
while $n < N$ and Left $\neq \{\phi\}$ **do**
  hub $\leftarrow \arg\max_{v \in \text{Left}} L(v)$
  spokes $\leftarrow \{i | (\text{hub}, i) \in \mathcal{E}\}$
  star $\leftarrow \{\text{hub}\} \cup \text{spokes}$
  $A \leftarrow A \cup \{\text{star}\}$
  Left $\leftarrow$ Left $\setminus$ star
  $n \leftarrow n + 1$
end while
output: set A of at most N query aspects

the qualifiers. Now, a query aspect is created out of the qualifier that is most frequent (called hub) in the historical data, and all the qualifiers connected to it (spokes). These nodes are deleted and the process continues until N query aspects have been generated or the set of available qualifiers becomes empty. Note that the set of aspects is mutually exclusive, but not necessarily exhaustive.

This is different from the original star clustering algorithm [1] in two ways. First, the aspects are now forced to be mutually exclusive, whereas in the original algorithm, only the hub nodes could not belong to multiple clusters. This was done so that the Pick-k algorithm could be used to efficiently pick the best query aspects for any query. Second, the original star clustering algorithm picks the highest *degree* nodes as hubs, whereas our approach uses the *frequency* of occurrence in the historical data. Hence, qualifiers picked as hubs are used for reformulating many different queries, or for very frequent queries, or both. This biases results towards the more important qualifiers; the query aspect stars built around such qualifiers should be frequently applicable to queries drawn from the query workload.

Empirical results in Section 4 demonstrate the strong performance of Algorithm 2. This might appear somewhat surprising, given the fact that the algorithm is not directly optimizing the objective function (Eq. 4). However, careful consideration of the algorithm shows that it is indeed *indirectly* optimizing the objective. Two qualifiers ℓ_1 and ℓ_2 are in the same query aspect $a \in A$ iff they have high cosine similarity to the hub qualifier ℓ_h, which happens iff both ℓ_1 and ℓ_h share many original queries, as do ℓ_2 and ℓ_h. But then, it is quite likely that ℓ_1 and ℓ_2 also share most of these queries. Hence, if a is among the best query aspects for such a query, then all three qualifiers ℓ_1, ℓ_2, and ℓ_h contribute significantly to the numerator of Eq. 5, and hence to the full objective function. Thus, the value of the objective will tend to be high, even though it is not explicitly being maximized. We can contrast this with other clustering algorithms such as single-linkage clustering, where there is no reason for nodes at opposite ends of the same cluster to have occurred for the same original query — indeed, the reverse is to be expected. Thus, it is the special structure of the star clustering algorithm that leads to the good results observed empirically, and this is the reason why we chose to extend the star clustering algorithm for our purposes.

Maximizing the objective directly. As noted above, star clustering optimizes the objective of Equation 4 only indirectly. We propose combining star clustering with a local search technique that directly increases the objective. Starting with the result of Algorithm 2, local search improves upon the solution in stages. In each stage, it finds a qualifier which can be moved from its current query aspect to a new one such that the total objective always increases. However, a straightforward application of this idea is infeasible: even a single local move could lead to a different set of top k query aspects being chosen for each query (the $\max$ in Equation 4), and recomputing these (using Algorithm 1) repeatedly would be very costly.

Our solution is based on the observation that the total objective does *not* need to be computed in each step: if the objective increases for a given assignment of k query aspects, it must necessarily increase with the optimal assignment of k aspects for each query. Hence, we first attempt to perform local moves without recomputing the best k aspects per query and only when such moves do not improve the objective do we perform the costly re-computations. This gives significant run-time savings. The space requirements are also low, as just k pieces of information need to be stored for each query (typically, $k \leq 5$). Should this become too large, we can work with only a query sample drawn according to their frequencies in the historical query workload.

4. EXPERIMENTS

The questions we want to answer in this section are: (a) How does our approach perform in user studies and automated evaluation as compared to baselines? (b) How does each component of our overall approach affect the final accuracy? (c) How does the performance of our approach change under different parameter settings and constraints? To answer these questions we conduct experiments on logs from real-world search engine traffic.

4.1 Experiment Design

We first describe our experimental design: datasets, evaluation measures, and baselines.

Evaluation data set. The data was generated from U.S. search query logs of a commercial search engine collected over a period of two months. These logs were processed to extract query *sessions*, as follows: For each day in the data set, the queries were grouped by users[3], ordered by time, and then broken up into sequences such that no two successive queries in any sequence were more than 10 minutes apart[4]. Each such sequence was defined to be an user session. While user session segmentation can be improved with more sophisticated algorithms (e.g., [16]), this simple low-cost heuristic performed adequately for our purposes.

Given user sessions, we needed to extract query reformulations. A query reformulation was defined as the following sequence of user actions within a session: (1) user types an original query q, (2) user does *not* click on any result, (3) user types another query q' such that q is a prefix of q', with a word/phrase r added as the suffix to q, and finally (4) user clicks on a result returned for the reformulated query. Now, r is defined as a *qualifier* for the *original query* q. Note that each session can yield multiple (q, r) pairs.

As a final step, the (q, r) pairs were split into a training set, derived from the first month of our search logs, and a testing set, derived from the first week of the second month's query logs. For both sets, all occurrences of (q, r) pairs were aggregated to yield triples of the form (q, r, count). All our experiments were run on this processed data.

From the logs in the training set, we selected the top 10,000 most frequent qualifiers (i.e., r) and their triples for constructing the set A of N query aspects. This serves to reduce both the computation complexity and the noise in the data, since qualifiers with low frequencies are too unreliable to be considered for broad query aspects. We used the logs in the test set to automatically evaluate the

[3]All user information such as user-IDs or IP addresses were removed.

[4]The 10-minute interval was picked after some exploratory data analysis; our methods can use any time interval.

performance of our approach. In order to get robust ground truth, we selected from the test set those original queries (i.e., q) whose accumulated counts of qualifiers are greater than 400; there are about 500 such cases. Each test case corresponds to a unique query, with its attendant qualifiers and frequencies.

Competing approaches and baselines. We compared two configurations of our approach against two baselines. In the MODSTAR configuration we obtain aspects A using the Modified-Star clustering procedure given in Algorithm 2 and employ Algorithm 1 to pick top k aspects from A for each test-set query. Our LOCSEARCH configuration is MODSTAR plus the use of the local search procedure in constructing A.

The first baseline, which we call ORGSTAR, employs the original Star clustering approach [1] to construct the final set of aspects A. In addition, the top k aspects to show to the user in response to a query are picked using Algorithm 1, as in MODSTAR.

As an additional point of reference, we consider a solution that has one query qualifier per aspect (BASELINE) and uses Algorithm 1 to pick the optimal set of k aspects to show to a user. The set of aspects A is constructed by picking the N most frequent single keywords in the historical data. Despite the seeming simplicity of BASELINE, it can be shown that, for any query q whose qualifiers $\ell(q)$ do not overlap with those of other queries, Algorithm 1 simply picks from among these N keywords the top k ones that occur most frequently in $\ell(q)$. Also note that as N grows large, the query aspects picked for each query are more and more tuned to that particular query and its qualifiers, yielding query suggestions [11] for that query in the limit. The fact that BASELINE is only a special case of our framework demonstrates its flexibility: At one end, we get broad multi-keyword query aspects that are globally optimized over the entire query workload, and at the other end, we can get locally optimal single-keyword query suggestions.

Evaluation criteria. We evaluated our approach using three different measures that measure orthogonal aspects of a user's experience with our system.

First we performed a manual post-hoc evaluation of the accuracy of the predicted aspects. The ground truth for this evaluation was obtained in a user study. Two human judges were each presented with 250 queries and their top 3 predicted aspects, and asked to rate the "goodness" of the individual aspects. Each aspect in a set of k aspects was rated individually as "good" if (a) the words and phrases belonging to the aspect were judged to form a coherent set, (b) the aspect was judged to be relevant to the query, and (c) the aspect was judged to be distinct from the other aspects. The output of our approach (LOCSEARCH) and BASELINE were judged this way, but the source of the predicted aspects was hidden from the judges. For uniformity of presentation, each aspect was given a name: For LOCSEARCH, the qualifier with the highest global frequency in the cluster was used as the aspect name, while for BASELINE, each aspect contained only one qualifier, which was used as the name. Using this data we define ACCURACY of an approach as the fraction of its predicted aspects that received "good" ratings.

The second relevant question we evaluated was: Are the predicted aspects consistent across similar queries? For example, we would like all queries related to, say, places and locations, to yield the same set of aspects and thus provide a consistent and predictable user experience. To measure this, we manually clustered the queries into 11 major semantic groups (plus one group for queries that did not fit into these categories) and computed the entropy of the predicted aspects in each group. Low ENTROPY in a query group implies that the set of predicted aspects is mostly the same, and is thus an indicator of consistency. Table 2 shows these 11 groups of queries.

ACCURACY and ENTROPY measure the precision and consistency of aspect predictions respectively. However, as pointed out in [11], it is difficult for a judge to accurately rate query suggestions without access to the context and task information inherent in a search session; the same is the case with judging predicted aspects. As a third measure of performance, we used click-through information to further validate the results obtained via the above measures. Our automatically obtained test data is in the form of triples (q, r, count), which reflect the number of times that users improve their query by appending r to query q. For any query q if we recommend aspect r with a high count, more users benefit from this recommendation. Thus we need to consider the actual count for a qualifier r in the evaluation. In Section 3.1 we showed how weighted F-measure captures the notion of usefulness of an aspect for a user query. In addition the F-measure in Equation 5 can take the actual count into account. Hence, in our experiments, we use the following two metrics: F@1 (weighted F-measure for the top recommended aspect) and F@3 (weighted F-measure for the top 3 recommended aspects). The normalization of qualifier sets $\ell(q)$ and aspects a_i are described in detail in Section 3.1.

4.2 Performance Comparisons

In this section, we compare the performance of different ways of recommending aspects. We first compare our proposed LOCSEARCH against BASELINE by manually evaluating the predictions of both. The results will demonstrate the superiority of the former in terms of both accuracy and non-redundancy of predicted aspects. Then we will present comparisons based on F@1 and F@3, which replicate these results, and also show that LOCSEARCH performs better MODSTAR and ORGSTAR.

For all these methods, we set the number of aspects to $N = 100$ and use Algorithm 1 to optimally pick aspects for each query. For ORGSTAR and MODSTAR, we set the similarity threshold to $\sigma = 0.25$.

Post-hoc manual evaluation. The manual evaluation was performed as a user study described in detail in Section 4.1. Averaged over all the aspects predicted for all the queries judged, the ACCURACY values for LOCSEARCH and BASELINE were 0.804 and 0.702 respectively. This indicates that LOCSEARCH achieves a 14% improvement over BASELINE in terms of ACCURACY. The two judges agreed with each other 90.5% of the times indicating that accuracy of LOCSEARCH approaches the upper-bound imposed by inter-rater agreement. Furthermore, for LOCSEARCH, at least 1 out of the 3 predicted aspects was deemed "good" a remarkable 98.4% of queries; all 3 predicted aspects were considered "good" 52% of the time. While BASELINE had a similar fraction of queries with at least one good prediction, only 28.6% of the queries elicited 3 good aspects. This is because (a) BASELINE can store only $N = 100$ query qualifiers, which is not enough, and (b) BASELINE often predicts redundant aspects, such as *pics* and *photos*.

The ACCURACY values for LOCSEARCH and BASELINE, broken down across query groups, are presented in Table 2; significant differences between the approaches are in bold. As we can see, LOCSEARCH significantly outperforms BASELINE in most query groups. *Place/Location* and *Male-Celebrities* are the two groups of queries where BASELINE outperforms LOCSEARCH. This is because these groups contained many queries that tended to have highly popular individual qualifiers that were not shared with other queries. For instance, while the location queries tended to have broad aspects such "maps" they also tended to have very popular specific qualifiers, like "cricket" for the query "india". Attempting to automatically discover when narrow query suggestions are useful in addition to broad aspects is a topic of further research.

Query category	Accuracy		Entropy	
	LOC-SEARCH	BASE-LINE	LOC-SEARCH	BASE-LINE
Companies with mostly online services (e.g., flickr, aol, bbc)	0.62	0.62	**3.59**	3.627
Companies with offline services including stores (e.g., walmart, fedex)	**0.64**	0.61	**3.147**	3.165
Products mostly sold/used/obtained online (e.g., photoshop, "get movies")	**0.75**	0.57	2.642	**2.246**
Products mostly sold/used/obtained offline (e.g., bmw, tattoos, cars)	0.70	0.71	**2.864**	3.361
Movies/Events/Shows (e.g., nascar, "kentucky derby")	**0.79**	0.72	**2.735**	2.884
Game related queries (e.g., sims, "gta 4", psp)	**0.94**	0.73	**2.471**	2.723
Showbiz characters (e.g., spongebob, batman)	**0.86**	0.67	**2.474**	2.734
Female celebrity names (e.g., "paula abdul", "madonna")	**0.86**	0.69	**1.894**	2.331
Male celebrity names (e.g., "tom cruise", "m jordan")	0.62	**0.78**	**1.94**	2.354
Place/Location (e.g., "las vegas", mexico)	0.56	**0.67**	**2.047**	2.282
Adult	**0.90**	0.78	**2.057**	2.815
None of the above	**0.80**	0.71	N/A	N/A

Table 2: Evaluation of LOCSEARCH versus BASELINE: LOCSEARCH has higher accuracy in predicting relevant aspects for a given query, and also better consistency (lower entropy) among aspects predicted for queries from the same query group. The last class is not used in entropy experiments since it is not a single semantic group.

Consistency of aspect predictions. As mentioned in Section 4.1 we would like similar queries to provide a consistent and predictable user experience. Table 2 lists the ENTROPY values for LOCSEARCH and BASELINE across the different query groups. As is evident, LOCSEARCH performs better than BASELINE on almost all query groups. In the case of some groups such as *Adult* and *Male-Celebrities*, the ENTROPY values are reduced by huge amounts, 24% and 18% respectively. In fact, the only reason the reductions in entropy are not more dramatic is because the entropy of aspects suggested by BASELINE is kept artificially low by redundancies in the predicted aspects. This is also the reason why BASELINE outperforms LOCSEARCH for the Online-Products category: BASELINE always predicts both "download" and "downloads" resulting in a low ENTROPY value. An interesting observation is that low ENTROPY values do not always indicate high user satisfaction. This is shown by the *Place/Location* and *Male-Celebrities* groups of queries where LOCSEARCH outperforms BASELINE in terms of ENTROPY but lags behind in terms of ACCURACY. Only ENTROPY and ACCURACY taken together give the complete picture.

Evaluation on click-through data. The hardness of our problem imposes limitations on the range of observable F-measure values: A high F-measure can be achieved only if the qualifiers of a large set of queries can be covered by picking only up to $k = 3$ aspects from a set of only $N = 100$ total aspects. As this is unlikely to be true in general, we need to establish a scale according to which to judge the resulting F-measure values. The best way would be to compare against the performance of the optimal N aspects, but we have no way to find these. However, we can find the best possible N *single-keyword* aspects by picking the N query qualifiers that occur most frequently in the *test set*. Even when the ground truth on the test set is known to the aspect selection algorithm, the maximum possible F-measure was found to be only 0.262. In the following, all reported F-measure values are normalized relative to this maximum value, and we call this the *Normalized F-measure*.

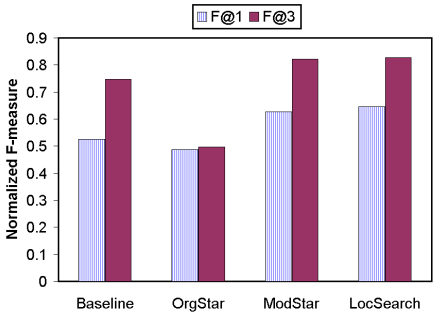

Figure 1: Performance comparison on F@1 and F@3.

Top word in aspect	Other top words in aspect			Example query
pictures	pics	photos	photo	bmw
download	downloads	freedownload	downlod	photoshop
video	videos	"video site"	playlists	"paula abdul"
lyrics	lyric	"song lyrics"	titles	madonna
games	game	"rpg games"	play	spongebob
movie	moive	"movie trailer"	movies	batman

Table 3: Top keywords of the most frequent aspects, and example queries for which they are predicted.

Figure 1 shows the results of BASELINE, ORGSTAR, MODSTAR, and LOCSEARCH at F@1 and F@3. Just as in the manual evaluation, LOCSEARCH outperforms BASELINE by a margin of 23% on F@1 and 11% on F@3. We also see that the clustering algorithm significantly affects result quality: Our proposed MODSTAR clustering performs much better than ORGSTAR, because it takes the qualifier *frequencies* into account, which biases the aspects towards the more important and common qualifiers. For example, in MODSTAR, the three frequent keywords "review," "reviews" and "rating" form a single aspect. However, ORGSTAR does not generate this aspect since this cluster is too small and none of the three keywords has a high enough degree. In fact, ORGSTAR performs worse than BASELINE, demonstrating that off-the-shelf clustering algorithms, applied directly to this problem, are unlikely to yield good results.

Sample aspects. Table 3 shows the most important word, and other top words, of the top 6 aspects discovered by our method, along with example queries for they are picked. The aspects are clearly coherent: All keywords in an aspect are semantically similar to the top word. All synonyms, mis-spellings, and singular/plural versions of an aspect keyword also belong to the same aspect. In addition, the aspects are all distinctive, each aspect describing a well-defined concept that is different from those of the other aspects. These aspects are also what one would expect to be the most common reformulations of Web queries. This demonstrates how our algorithms achieve the right balance between query coverage and aspect specificity.

4.3 Parameter Settings

We now study the impact of different parameters on the performance of our algorithms. In particular, we consider the effects of the similarity threshold σ used in MODSTAR and LOCSEARCH, and the limit on the number of query aspects N.

The similarity threshold σ. The MODSTAR algorithm (and consequently, LOCSEARCH as well) merges query qualifiers into an aspect cluster using a similarity threshold parameter σ. In Figure 2,

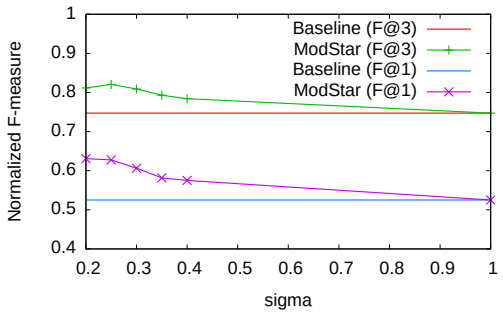

Figure 2: The impact of similarity threshold parameter σ on F@1 and F@3. When $\sigma = 1$, MODSTAR reduces to BASELINE.

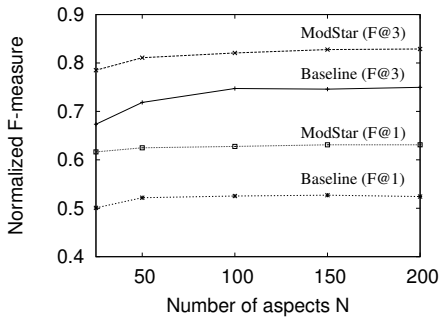

Figure 3: The impact of number of aspect N on F@1 and F@3. The curves flatten out by $N = 100$.

we investigate the sensitivity of the F-measure to σ. We also show the result of BASELINE for comparison. As σ increases, fewer qualifiers are considered to be similar, with each qualifier forming its own cluster in the limit of $\sigma = 1$. Thus, $\sigma = 1$ is equivalent to BASELINE, as can be observed from the F-measure results. On the other hand, when σ is too small, each aspect becomes large and thus less coherent, leading to low F-measure values. The best performance is obtained when we set $\sigma = 0.25$. We use this value in all other experiments.

Number of Aspects N. We also study the impact on performance of the number of aspects N. Figure 3 shows the results of BASELINE and MODSTAR when we vary N from 25 to 200, for both F@1 and F@3. Clearly, when we enlarge the number of aspects N, all the methods are improved. This is because with increasing N, the set of aspects contains more query qualifiers and can thus yield good predictions for more test queries. In addition, the aspects themselves can be smaller and hence more coherent. The behavior of LOCSEARCH is similar, and is not shown in the interest of clarity. The plots flatten out by $N = 100$, implying that a relatively small set of aspects can be sufficient to cover most web queries. This is because the frequency with which any given aspect is recommended in response to a test query is highly skewed: The "pictures" aspect is recommended for over 40% of the queries while there are 60 different aspects that are recommended for less than 2% of the queries. Thus, a total of $N = 100$ aspects provide adequate performance, and this is the value we use in the other experiments.

5. CONCLUSIONS

A common problem with user queries is that they are underspecified, or that they have senses that the user did not think of. A broad latent query aspect is a set of keywords that succinctly represents one particular sense, or one particular information need, that can aid users in reformulating such queries. We extract these query aspects from query reformulations obtained from user session logs; since reformulations are exactly what the extracted query aspects are then used for, historical query reformulations are particularly well-suited for our problem domain.

We propose a new optimization-based formulation and algorithms for extracting such broad latent aspects from historical user session logs via an offline process. We also present an optimal and efficient algorithm to pick, at run-time, the top k query aspects to be shown to the user in response to her search query. Empirical studies on a large real-world corpus of search logs show significant gains over a strong baseline that uses single-keyword reformulations: a 14% gain in accuracy of aspects as judged by human editors, around 20% improvement in consistency among aspects predicted for "similar" queries, and 23% gain in terms of weighted F-measure. All of these demonstrate the importance and accuracy of the broad query aspects that we find.

6. REFERENCES

[1] J. A. Aslam, E. Pelekov, and D. Rus. The star clustering algorithm for static and dynamic information organization. *Journal of Graph Algorithms and Applications*, 8(1):95–129, 2004.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, New York, USA, 1999.

[3] B. Billerbeck, F. Scholer, H. E. Williams, and J. Zobel. Query expansion using associated queries. In *CIKM*, 2003.

[4] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *CIKM*, 2008.

[5] C. Buckley and G. Salton. Optimization of relevance feedback weights. In *SIGIR*, pages 351–357, 1995.

[6] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *KDD*, 2008.

[7] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

[8] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *WWW*, 2005.

[9] P. A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *SIGIR*, pages 7–14, 2007.

[10] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM*, 2008.

[11] S. Cucerzan and E. Brill. Extracting semantically related queries by exploiting user session information. `http://research.microsoft.com/users/silviu/Papers/np-www06.pdf`.

[12] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *WWW*, pages 325–332, 2002.

[13] B. M. Fonseca, P. Golgher, B. Pôssas, B. Ribeiro-Neto, and N. Ziviani. Concept-based interactive query expansion. In *CIKM*, 2005.

[14] A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal. Using the wisdom of the crowds for keyword generation. In *WWW*, 2008.

[15] B. J. Jansen, A. Spink, and J. Pedersen. A temporal comparison of altavista web searching: Research articles. *J. Am. Soc. Inf. Sci. Technol.*, 56(6):559–570, 2005.

[16] R. Jones and K. Klinkner. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *CIKM'08*.

[17] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW*, pages 387–396, 2006.

[18] M. Pasca and B. V. Durme. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *ACL*, pages 19–27, 2008.

[19] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1), 1999.

[20] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, April 2005.

[21] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD*, pages 131–142, 2004.

[22] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *WWW*, pages 11–18, 2003.