Organizing Query Completions for Web Search

Alpa Jain
Yahoo, CA, USA
alpa@yahoo-inc.com

Gilad Mishne
Yahoo, CA, USA
gilad@yahoo-inc.com

ABSTRACT

All state-of-the-art web search engines implement an auto-completion mechanism—an assistive technology enabling users to effectively formulate their search queries by predicting the next characters or words that they are likely to type. Query completions (or suggestions) are typically mined from past user interactions with the search engine, e.g., from query logs, clickthrough patterns, or query reformulations; they are ranked by some measure of query popularity, e.g., query frequency or clickthrough rate. Current query suggestion tools largely assume that the set of suggestions provided to the users is homogeneous, corresponding to a single real-world interpretation of the query. In this paper, we hypothesize that, in some cases, users would benefit from an alternative presentation of the suggestions, one where suggestions are not only ordered by likelihood but also organized by high-level user intent. Rich search suggestion interaction frameworks that reduce the user effort in identifying the set of relevant suggestions open new and promising directions towards improving user experience. Along these lines, we propose clustering the set of suggestions presented to a search engine user, and assigning an appropriate label to each subset of suggestions to help users quickly identify useful ones. For this, we present a variety of unsupervised clustering techniques for search suggestions, based on the information available to a large-scale web search engine. We evaluate our novel search suggestion presentation techniques on a real-world dataset of query logs. Based on a set of user studies, we show that by extending the existing assistance layer to effectively group suggestions and label them—while accounting for the query popularity—we substantially increase the user's satisfaction.

Categories and Subject Descriptors

H.5.0 [**Information Interfaces and Presentation**]: General

General Terms

Algorithms, Experimentation

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

1. INTRODUCTION

Search engines are increasingly exploring ways to reduce user efforts in performing search-related tasks. Such efforts have resulted in the widely-used auto-completion mechanism that automatically suggests possible completion of search queries while users are formulating their queries.

EXAMPLE 1. *Consider the case where a user initiates a search engine query by typing the character sequence, 'haifa'. For this query prefix, the set of suggestions presented by a major search engine could include the following ordered list:*

haifa
haifa wehbe
haifa wahbi
haifa israel
haifa photo
haifa hot
haifa university

Various factors such as, click behavior, query frequencies, or query reformulations, based on past user behavior determine the set of query completions offered by a search engine. This paper extends the current query completion approach by *organizing auto-complete suggestions by topic.*

The above example underscores two important observations which form the basis of our work. First, query completion suggestions may correspond to non-identical real-world entities or facets. For instance, the suggestions at position 1, 2, and 5 relate to a popular entertainer, whereas the suggestions at position 3, and 6 relate to a city. Second, query suggestions associated with similar facets may not be grouped together and thus, suggestions may often be an unordered list from a topicality perspective. Contrast the presentation above with the following alternative presentation:

haifa	
haifa wehbe	Haifa (Singer)
haifa wahbi	
haifa photo	
haifa hot	
haifa israel	Haifa (City)
haifa university	

Our goal is to identify (implicit) topics or facets among a set of suggestions for a query prefix, and organize and present suggestions in a topic-aware manner as shown in the above example. For this several problems need to be addressed. First, we need to identify appropriate representations for each suggestion for a prefix such that they capture

user perception. For instance, while "walmart pharmacy" and "walmart careers" may be related to a single real-world concept, some users may perceive these as related but dissimilar concepts. Second, we need to enable quick identification of desired clusters while keeping the user efforts low. For instance, in the above example, providing a representative image or textual label for each suggestion cluster can help users distinguish between the clusters and locate one that matches their needs. Third, this novel setting of presenting clusters of suggestions, instead of suggestions, introduces a new ranking challenge of deciding how to order suggestion clusters for a query prefix.

In this paper, we present an end-to-end approach for automatically *organizing and presenting completions for web search queries*. Specifically, our contributions are:

- A set of techniques to automatically cluster suggestions for a query prefix that can help users quickly identify desired suggestions.
- A set of methods to identify discriminative labels for each cluster of suggestions.
- A placement algorithm to present clusters with the goal of minimizing user effort in locating desired suggestions.
- An extensive experimental evaluation over real-life datasets, including user studies demonstrating the proposed problem of clustering suggestions for query prefixes.

The rest of the paper is structured as follows: Section 2 discusses the problem on which we focus and presents methods to characterize and cluster similar suggestions for a query prefix. Section 3 discusses our methods to identify appropriate labels to the clusters generated for a set of suggestions. Section 4 discusses our method to measure user effort which, in turn, leads to a ranking algorithm to order suggestion clusters. Section 5 reports the results of our experimental evaluation. Finally, Section 6 reviews related work, and Section 7 concludes the paper.

2. CLUSTERING SUGGESTIONS BY TOPIC

A *query prefix* (or *prefix*) is a sequence of characters typed by a users while formulating her query. Given a prefix p, a search engine returns an ordered set S of suggestions for completing the query that started with p.

PROBLEM 2.1. *Given a prefix p and an ordered set of suggestions, $S = \{s_1, s_2, \ldots, s_n\}$, our goal is to partition S into k ordered, disjoint partitions, $P = \{P_1, P_2, \ldots, P_k\}$, such that every s_i belongs to exactly one P_j, and the members of every P_j are* topically-coherent [1] *, i.e., refer to a single topic or aspect of q. After partitioning S, we wish to assign a distinct label L to each partition such that $\mathrm{L}(P_j)$ describes to a user that topic which is shared by members of P_j, but not by the rest of the elements in S. Finally, we wish to rank the partitions $\mathrm{L}(P_j)$ as well as the suggestions within each partitions so as to maximize the* utility *(see Section 4) of the set S to the user.*

[1] Alternatively, partitions may be defined along other dimensions (e.g., transactional or navigational queries); in this paper, our primary goal is to assist users in quickly parsing available query suggestions and therefore we begin with intuitive and easily recognizable groupings, i.e., by topics.

To address this problem, we begin by examining three different approaches to the task of clustering suggestions for a query prefix. Our approaches make use of the search engine itself, with an increasing level of dependency. We start with a clustering mechanism requiring only the number of documents returned by the search engine for a given query(Section 2.1); continue with a different approach utilizing the full text of the search results (Section 2.2); and discuss a clustering technique that employs implicit user feedback by examining the documents clicked by users issuing the query (Section 2.3).

2.1 Head Word Clustering

Many of the suggestions offered as the users types are completions, treating the user input as a prefix (and, sometimes, a suffix or infix). As a result, the set S is usually already very similar at the lexical level. In general, a suggestion s_i can be viewed as $s_i = p \cup c_i$, where p is the user-supplied query prefix, and c_i is additional context added in the particular suggestion s_i. For example, consider the top suggestions for $p =$ `salsa`:

salsa
salsa recipes
salsa dancing
salsa dance
salsa music
salsa singer cruz
homemade *salsa*
salsa lessons
salsa classes

The different suggestions to be clustered already share p; the terms that may be useful for identifying the cluster s_i belongs to are more likely in c_i. We hypothesize that we can select a single term from each s_i—the *most discriminative* term—and that clustering these terms only will translate to a good clustering on S itself. In the example above, the discriminative terms are *recipes*, *dancing*, *dance*, and so on. We refer to such a term as the *head word* of s_i.

Once the clustering task is reduced from the query level to the term level, we can employ a multitude of existing approaches for estimating semantic or topical word-level similarity. Commonly-used methods include those based on word contexts in a large corpus or lexical resources such as Wordnet [18, 6]. We choose *PMI-IR*, a simple co-occurrence technique shown to be effective in similar settings [25, 5]. Here, the similarity between two words $\{w_i, w_j\}$ is defined as the pointwise mutual information between the words, where the probability of a single word, $P(w_i)$, as well as the joint probability $P(w_i, w_j)$ are estimated using maximum likelihood of occurrences in a corpus. Specifically, the similarity measure between the words in this case is defined as

$$Sim(w_i, w_j) = \log \frac{\frac{|hits(w_i) \cap hits(w_j)|}{n}}{\frac{|hits(w_i)|}{n} \cdot \frac{|hits(w_j)|}{n}} \quad (1)$$

where $hits(x)$ is the set of documents containing x and n is the corpus size. As mentioned earlier, our only requirement form the search engine for this approach is obtaining $hits(x)$; we set the similarity between suggestions to be the similarity between their head words.

Head Word Selection

Due to the short average length of web queries, c_i often consists of a single term only. However, there are cases where

the head word needs to be chosen from several candidates. We experiment with several simple approaches to selecting the head word:

- **First word.** Select the leftmost word in c_i, e.g., *singer* in `salsa singer cruz`.
- **Last word.** Select the rightmost word in c_i, e.g., *cruz* in `salsa singer cruz`.
- **Frequency.** Compute, for each word in c_i, its *tf·idf* value, where the "document" for computing *tf* consists of all words in the suggestion set S being clustered, and *idf* is computed over the set $\mathcal{S}$ of all suggestions for all queries:

$$tf(w) = \frac{\sum_{s \in S} \text{count}_w(s)}{\sum_{s \in \mathcal{S}} |s|} \quad , \quad idf(w) = \log \frac{|\mathcal{S}|}{|\{s | w \in s\}|}$$

  Then, select the word with the highest value.

2.2 Result-set Clustering

In the context of web search, a natural approach to computing the similarity between two queries (or query suggestions) is to leverage the search results associated with each. Existing work in this direction [23, 27] represents queries using *tf·idf*-weighted term vectors of frequent terms found in the top-N search results for the query; cosine similarity between these vectors is shown to be effective as a query similarity measure. Our next proposal to clustering query suggestions largely follows this approach, extending it slightly by separately utilizing terms appearing in the titles of top-ranked documents, their URLs, and the content of the documents themselves.

Concretely, given a query suggestion s we obtain the set $R(s)$ of the top-n documents for s returned by a search engine. Each document $d \in R(s)$ contains a title, a URL, and an "abstract" – a snippet of d that is shown to the user, containing the terms in the query and a small amount of context around them. We mark these as $t(d)$, $u(d)$, and $a(d)$, respectively. Next, we construct term vectors for each of these components in a manner similar to [23]: each document component is represented using a *tf·idf* vector of the terms appearing in it, and the result set as a whole is represented using the centroid of these vectors. We concatenate the vectors formed by $t(d)$, $u(d)$, and $a(d)$ to obtain a single vector representing the result set, $\overrightarrow{v_s}$. [2] The similarity function we then use to cluster the suggestions in this method is the inner product between the vectors representing the result sets of the suggestions,

$$Sim(s_i, s_j) = \overrightarrow{v_{s_i}} \cdot \overrightarrow{v_{s_j}} \tag{2}$$

2.3 Click-based Clustering

Our third approach leverages *clickthrough data* maintained by search engines, which contains information about urls from the search results presented to the users that were clicked. For instance, a search log may contain the following clicked urls for a query *pineapple salsa*, for different users:

u_1: `www.allrecipes.com/pineapple-salsa/detail.aspx`
u_2: `www.cooks.com/rec/pineapple_salsa.html`
u_3: `www.blogchef.net/pineapple-salsa-recipe/`

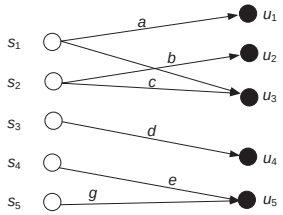

Figure 1: Sample bipartite clickthrough graph.

Using the clickthrough data, we can characterize each suggestion for a prefix by the set of clicked urls associated with it and group suggestions with similar user click behavior. The intuition behind this clustering method is that non-identical queries that generate clicks on the same urls capture similar user intent. For instance, the query *pineapple salsa for fish* may also generate clicks on the one of the above urls, indicating that the two suggestions are similar.

Before representing query suggestions using clickthrough data, we discuss two main observations. First, using clicked urls as is could result in *specific* representations which prove to be too restrictive since, websites tend to dedicate a web page per concept. So, we generalize our suggestion representation by using base urls from the clickthrough data. For instance, the url clicked by user u_1 is generalized to `www.allrecipes.com`. Second, encyclopedic websites such as, `www.wikipedia.org`, may introduce undesired bias and lead to non-similar concepts to be placed in the same cluster. For instance, we observed the most frequently clicked base url for both *gold retriever* and *gold rush 1849* is `www.wikipedia.org`. Similarly, other websites such as, `wwww.youtube.com` may also introduce such bias. To address this issue, we can treat each suggestion as a "document" and compute an *inverse document frequency* for each base url and use that as the weight when generating the representation. In our experiments, we employed a stop-list by eliminating top-5 urls based on their inverse document frequency, which tends to work better than using the inverse document frequency as weights.

Given a prefix p and the set S of suggestions associated with it, we define a *clickthrough graph* for p.

DEFINITION 2.1. **[Clickthrough graph]** *A clickthrough graph is a bipartite, directed graph consisting of two classes of nodes:* suggestions *nodes ("s" nodes) and* base urls *nodes ("u" nodes), and a set of directed edges E. Each suggestion in the set S is represented as an s node. To generate the u nodes, we take the union of the set of base urls associated with each suggestion and generate a node per distinct base url. An edge $s \rightarrow u$ between a suggestion node s and url node u indicates that url u was clicked when s was issued as a query. Each edge is assigned a weight which is the number of times u was clicked when s was issued as a query.* □

Using the clickthrough graph, for each suggestion s in the graph, we generate a L2-normalized feature vector of size equal to the number of url nodes in the graph, and each dimension in the vector represents a url in the graph. The value for the dimension associated with url j is computed as:

$$\vec{f_j} = \begin{cases} \frac{w_{sj}}{\sqrt{\sum_i^{|\mathcal{U}|} {w_s i}^2}} & \text{if exists an edge between suggestion } s \text{ and } j; \\ 0 & \text{otherwise.} \end{cases}$$

where $\mathcal{U}$ is the set of urls in the clickthrough graph and w_{sj}

[2] Note that a word may be represented more than once in the final term vector if it appeared in different components of the document, i.e., in the title and the URL. In practice, this is equivalent to prefixing each term with its component (e.g. *title.word*) before constructing the term vector.

is the weight associated with edge $s \rightarrow j$ in the clickthrough graph. To compute the similarity between two suggestions for a prefix, we experimented with two similarity functions and picked cosine-similarity defined as:[3]

$$Sim(x, y)) = \sum_{i}^{|\mathcal{U}|} \frac{x_i \cdot y_i}{\sqrt{\sum_i {x_i}^2} \cdot \sqrt{\sum_i {y_i}^2}} \quad (3)$$

Clustering Algorithm

So far, we discussed three different way of characterizing suggestions for a query prefix, varying in the way topics are modeled; each of these results in a different similarity measure between suggestion. Once this similarity is estimated, performing the clustering itself is straightforward; we use Hierarchical Agglomerative Clustering to partition the suggestion set into individual clusters using the similarity.

3. LABELING CLUSTERS

User studies on clustered presentation of web search results consistently show that users prefer clusters that are assigned a meaningful title over clusters with no label [14]. We hypothesize that the same holds for a clustered presentation of query suggestion. In this section, we describe several methods for assigning a meaningful label to a set of suggestions. To demonstrate these approaches over different suggestion sets, we refer to two query prefixes and their (clustered) suggestions appearing in Figure 2.

	`los an`	`nursi`	

LA.1	`los angeles daily news`
	`los angeles times`
	`los angeles times newspaper`
		`nursing`
		`nursing jobs`
		`certified nursing`
			NU.1

LA.2	`los angeles public library`
	`los angeles police department`
	`los angeles unified school district`
		`nursing homes`
		`nursing home compare`
		`nursing home costs`
			NU.2

LA.3	`los angeles lakers`
	`los angeles dodgers`
	`los angeles angels`
		`nursing scrubs`
		`nursing shoes`
		`nursing uniforms`
			NU.3

Figure 2: Sample suggestion clusters; a cluster identifier is shown next to each cluster.

Most Frequent Suggestion (MFS): One way to select a label for a cluster of query suggestions is to select the most representative suggestion in the cluster. Since every suggestion is a query by itself, a natural way to select the most representative suggestion is to choose the most frequent one in the search engine's query log. Formally, the label assigned by MFS to the set of suggestions S is

$$\text{MFS}(S) = s_i : s_i \in S,\ \forall_{s_j \in S}\ \text{Freq}(s_j) \leq \text{Freq}(s_i)$$

Where $\text{Freq}(x)$ is the number of times x is observed in a large query log. In the examples appearing in Figure 2, cluster *LA.1* would be assigned the label `los angeles daily news` using this method; cluster *NU.1* would be assigned `nursing`; and so on.

Longest Common Subsequence (LCS): Often, a sequence of characters is shared among all suggestions within a cluster, but not with suggestions in other clusters; for example; the user-supplied "us a" may be completed to *us airways, us airways flights, ...* as well as to *us army, us army jobs, ...*, and so on. We hypothesize that in some cases it is beneficial to use the *longest common subsequence* of the suggestions as the label. Formally, the label assigned by LCS to S is

$$\text{LCS}(S) = l_i : l_i \in \text{Q(S)},\ \forall_{l_j \in Q(S)}\ \text{Length}(l_j) \leq \text{Length}(l_i)$$

Where $Q(S)$ is the set of subsequences of any $s \in S$. For example, the label assigned by this method to clusters *LA.1* through *la.3* is `los angeles`, whereas the label assigned to cluster *NU.2* is `nursing home`.

Most Frequent in Result Set (MFRS): One drawback of both MFS and LCS is that they always draw a label from the suggestions belonging to the cluster. For some clusters of suggestions, a meaningful label is not part of the cluster and has to be obtained using external resources; for example, for the cluster *LA.1* in Figure 2, a useful label may be `los angeles newspapers` – a label that has only partial overlap with the suggestions in the cluster.

As with the case of performing the clustering itself, we turn to the top-ranked documents for each suggestion (when it is used as a query to a search engine) for this external knowledge. By transforming the suggestion set into a document set we can use a variety of methods developed for labeling documents, rather than queries. We adopt a standard approach to labeling clusters of documents, namely, harvesting word n-ngrams from them and selecting the most frequent n-gram [9]. Formally, let $R(s)$ be the set of top-ranked results for the suggestion s; let $R(S) = \cup_{s_i \in S} R(s_i)$; let $NG(d)$ be the set of word n-ngrams contained in the document d; and let $NG(R(S))$ be the set of all n-grams in all top-ranked documents, $NG(R(S)) = \cup_{d \in R(S)} NG(d)$. Then the label assigned by MFRS to S is

$$\text{MFRS}(S) = l_i : l_i \in \text{NG(R(S))},\ \forall_{l_j \in NG(R(S))}\ \text{Count}(l_j, R(S)) \leq \text{Count}(l_i, R(S))$$

This method assigns, for example, the label `news` to cluster *LA.1*, and the label `nursing jobs` to *NU.1*.

Most Frequent in Modified Result Set (MFRS*): Finally, we return to our observation from Section 2.1: search suggestions are unique as a collection of entities to cluster in that they often have a high degree of lexical overlap. In a cluster with a long common subsequence (such as *LA.2*), the elements we are interested in labeling are sometimes best represented in those portions of the suggestions that are not shared among all elements of the cluster (e.g. *public library, police department*). To this end, we propose an additional labeling mechanism, similar to MFRS, but where the queries we use are not the suggestions themselves, but the portions of the suggestions that are distinct within the cluster. Formally, let s_i^* be the suggestion s_i with the longest common subsequence of S removed, $s_i^* = s_i - \text{LCS}(S)$, and let S^* be the set of suggestions in S with the longest common subsequence removed from all suggestions, $S^* = \cup_i\ s_i^*$, then the label assigned by MFRS* to S is

$$\text{MFRS}^*(S) = \text{MFRS}(S^*)$$

For example, this method assigns the label `services` to the cluster *LA.2*.

Combined Labeling Strategy (Comb): As seen in the examples in 2, clusters of suggestions have different characteristics, and may benefit from different labeling approaches.

[3]We experimented with Euclidean distance but observed worse performance than that for cosine-similarity. We omit details from other distance functions due to space constraints.

```
RankClusters(S, f)
Input: Set S of clusters C, Map f of suggestions and their
       frequencies
Output: Ordered list L of clusters in S
F = ∅
foreach cluster C ∈ S do
    F(C) = 0
    foreach suggestion s ∈ C do
        F(C) += f(s)
    end
    Rank suggestions in C by their frequency using f
end
Rank clusters in S based on their aggregate frequency F(C) and
generate L
return L
```

Figure 3: Algorithm to order clusters to minimize the expected cost.

Our final labeling is a hybrid one, first selecting which labeling method to use, then assigning the label itself;. We refer to this method as Comb.

The main difference between our labeling approaches is whether the label is assigned from within the cluster (MFS, LCS), or from external knowledge (MFRS, MFRS*). The Comb approach selects among these by examining the cluster *cohesion*, the degree to which the cluster elements are similar; the more compact a cluster is, the more likely it is that a good label is found in its members rather than externally. The cohesion of S is measured using the average distance between the elements of S: Cohesion$(S) - \frac{1}{|S|}\Sigma_{i,j} Sim(s_i, s_j)$; the label assigned by Comb is then

$$\text{Comb}(S) = \begin{cases} \text{MFS}(S) & \text{if Cohesion}(S) < k \\ \text{MFRS}(S) & \text{otherwise.} \end{cases}$$

4. RANKING SUGGESTION CLUSTERS

Our objective in clustering suggestions is to reduce the users' effort in locating their desired query completions. Naturally, the order in which suggestions are presented influences the amount of user effort. With this in mind, we describe a cost metric to characterize the user effort spent in locating a suggestion from among a set of clusters of suggestions. We then present an algorithm that minimizes the expected cost of locating suggestions.

We begin by describing a structural property of clusters of suggestions fundamental to our cost metric. By clustering (and labeling) the set of suggestions available for a prefix, we are generating a *skip list* of the suggestions where users will first skip between buckets (i.e., clusters) and then upon identifying a relevant bucket, the user will scan within the bucket to locate the desired suggestion. Thus, the cost of identifying a suggestion s consists of:

- *Time to read a cluster label:* Given a prefix users begin with browsing the clusters of suggestions for the prefix by reading the labels. At each cluster C, the user decides if the cluster should be skipped or scanned, depending on whether the label captures the user's area of interest. We denote the cost in reading the label of a cluster C as $T_{lb}(C)$.
- *Time to scan a cluster:* Once a cluster C that contains the desired suggestion s has been identified, users scan suggestions in C until s is located. We denote the cost of scanning each suggestion s in the cluster by $T_{sc}(s)$.

Consider a user who has provided a prefix p and is interested in locating suggestion s from a set of clusters $C_1, C_2 \cdots, C_n$, and let C_m be the cluster than contains suggestions $s_1, s_2, \cdots, s_{|Cm|}$ such that $s_k = s$ i.e., s is located at position k within C_m. The cost of locating suggestion s for the user, which we denote $T(s)$, is then $\sum_{i=1}^{m} T_{lb}(C_i) + \sum_{j=1}^{k} T_{sc}(s_j)$. For simplicity, we assume that the cost to read any cluster label is the same for all clusters, namely T_{lb}. Similarly, we assume the cost to scan through suggestions within a cluster is T_{sc}, the same regardless of the suggestion. $T(s)$ for a suggestion s at position k in cluster m then becomes simply

$$T(s) = m \cdot T_{lb} + k \cdot T_{sc} \tag{4}$$

For a user who has entered prefix p, we would like to study the expected cost $T(p)$ of locating the suggestion she is interested in among all the suggestions provided. If we denote by $P\{s|p\}$ the probability that the user prefers suggestion s when she entered prefix p then the expected cost is:

$$T_p(R) = \sum_{\forall s} T(s) \cdot P\{s|p\} \tag{5}$$

In our notation we mark T_p as a function of the ranking R of our suggestion, to emphasize that the cost is very much dependent on the ranking R. $P\{s|p\}$ can be estimated from the query logs based on observed user preferences when entering prefix p. Specifically, if $f(p)$ is the number of times the prefix p was entered, and $f(s)$ is the number of times suggestion s was submitted as a user query, then:

$$P\{s|p\} = \frac{f(s)}{f(p)} \tag{6}$$

Note that $\sum_{\forall s} P\{s|p\}$ in general will be less than 1, as users may have entered queries that are not among our suggestions list. We assume the cost to the user interested in a suggestion not present in our list to be independent of the ranking of the list of suggestions that are present, and we don't count it as part of T_p.

The goal of our ranking algorithm is then to order the clusters and suggestions such as to minimize $T_p(R)$. Figure 3 shows the algorithm we use to find the optimal ordering of suggestions R. Specifically, the algorithm ranks suggestions withing a cluster in nonincreasing order of their frequencies $f(s)$. To rank clusters of suggestions, each cluster C is assigned and aggregate frequency $F(C)$ equal to the sum of the frequencies of all suggestions in C. We show that Algorithm 3 generated a ranking R that minimizes the cost $T_p(R)$ using the following Theorem.

THEOREM 4.1. *Given a set of clusters S, RankClusters generates an optimal ranking R that minimizes the total expected cost of finding a suggestion in S.*

PROOF. By contradiction.

Case 1: Ranking within a cluster. Assume that there exists an optimal ranking R such that a cluster C_m contains suggestions s_x and s_y at positions x and y. Assume for contradiction that $f(s_x) > f(s_y)$, but $x > y$. We will show that by swapping s_x and s_y in C_m, we achieve a new ranking R' with total cost is less than that of R, thereby contradicting the optimality of R. Based on Equation 5 we can express $T_p(R)$ as $\sum_{\forall s!=x,y} T(s) \cdot P\{s|p\} + T(s_x) \cdot P\{s_x|p\} + T(s_y) \cdot P\{s_y|p\}$. Similarly, $T_p(R')$ as $\sum_{\forall s!=x,y} T'(s) \cdot P\{s|p\} + T'(s_x) \cdot Ps_x|p + T'(s_y) \cdot P\{s_y|p\}$. First, we note that by swapping s_x and s_y, we have not altered the cost of any suggestion other than s_x and s_y. Therefore, $T(s) = T'(s)$ for any $s! = x, y$. We can therefore compute $T_p(R) - T_p(R') =$

$T(s_x)\cdot P\{s_x|p\}+T(s_y)\cdot P\{s_y|p\}-[T'(s_x)\cdot P\{s_x|p\}+T'(s_y)\cdot P\{s_y|p\}] = [T(s_x)-T'(s_x)]\cdot P\{s_x|p\}+[T(s_y)-T'(s_y)]\cdot P\{s_y|p\}$. Using 4 and 6, $T_p(R)-T_p(R')$ becomes $[m\cdot T_{lb}+x\cdot T_{sc}-(m\cdot T_{lb}+y\cdot T_{sc})]\cdot f(s_x)/f(p)+[m\cdot T_{lb}+y\cdot T_{sc}-(m\cdot T_{lb}+y\cdot T_{sc})]\cdot f(s_y)/f(p) = 1/f(p)\cdot[T_{sc}\cdot(x-y)\cdot f(s_x)+T_{sc}\cdot(y-x)\cdot f(s_y)] = 1/f(p)\cdot T_{sc}\cdot(x-y)\cdot[f(s_x)-f(s_y)] > 0$ according to our assumptions. This shows that $T_p(R) > T_p(R')$, proving our contradiction.

Case 2. Ranking of clusters. Assume that there exists an optimal ranking R such that a cluster C_x with aggregate frequency $F(C_x)$ appears at position x, and a cluster C_y with aggregate frequency $F(C_y)$ appears a position y. Assume for the sake of contradiction that $F(C_x) > F(C_y)$ but $x > y$. We will show that by swapping C_x and C_y we achieve a new ranking R' whose total cost is less than R, thereby contradicting the optimality of R. Based on Equation 5 we can express $T_p(R)$ as $\sum_{\forall s\notin C_x,C_y} T(s)\cdot P\{s|p\}+\sum_{s\in C_x} T(s)\cdot P\{s|p\}+\sum_{s\in C_y} T(s)\cdot P\{s|p\}$. Similarly, $T_p(R') = \sum_{\forall s\notin C_x,C_y} T'(s)\cdot P\{s|p\}+\sum_{s\in C_x} T'(s)\cdot P\{s|p\}+\sum_{s\in C_y} T'(s)\cdot P\{s|p\}$. As before, we note that the cost of any suggestion not in C_x or C_y is not affected by swapping C_x and C_y and therefore $T(s) = T'(s)$ for any $s \notin C_x, C_y$. We can then compute $T_p(R)-T_p(R') = \sum_{s\in C_x} T(s)\cdot P\{s|p\}+\sum_{s\in C_y} T(s)\cdot P\{s|p\}-[\sum_{s\in C_x} T'(s)\cdot P\{s|p\}+\sum_{s\in C_y} T'(s)\cdot P\{s|p\}] = \sum_{s\in C_x}[T(s)-T'(s)]\cdot P\{s|p\}+\sum_{s\in C_y}[T(s)-T'(s)]\cdot P\{s|p\}$. Using 4 and 6, this becomes $\sum_{s_k\in C_x}[x\cdot T_{lb}+k\cdot T_{sc}-(y\cdot T_{lb}+k\cdot T_{sc})]\cdot\frac{f(s_k)}{f(p)}+\sum_{s_l\in C_y}[y\cdot T_{lb}+l\cdot T_{sc}-(x\cdot T_{lb}+l\cdot T_{sc})]\cdot\frac{f(s_l)}{f(p)}$

$= \sum_{s_k\in C_x}(x-y)\cdot T_{lb}\cdot\frac{f(s_k)}{f(p)}+\sum_{s_l\in C_y}[(y-x)\cdot T_{lb}\cdot\frac{f(s_l)}{f(p)}$
$= (x-y)\cdot T_{lb}\cdot 1/f(p)\cdot[\sum_{s\in C_x} f(s)-\sum_{s\in C_y} f(s)] = (x-y)\cdot T_{lb}\cdot 1/f(p)\cdot[F(C_x)-F(C_y)] > 0$ based on our assumption that $x-y>0$ and $F(C_x)-F(C_y)>0$. This shows that $T_p(R) > T_p(R')$, proving our contradiction. ∎

5. EXPERIMENTAL EVALUATION

In this section, we present our experimental evaluation. We first discuss our data sets(Section 5.1) and our observations from two pilot studies examining whether users can benefit by the problem solved in this paper (Section 5.2). Then, we evaluate our suggestion clustering techniques (Section 5.3), cluster labeling techniques (Section 5.4), and techniques to order suggestion clusters (Section 5.5). We conclude in Section 5.6.

5.1 Data Sets

Query logs: We collected a random sample of 100 million, fully anonymized queries sent to the Yahoo! web search engine in the first seven months of 2009, along with their frequency. We sort these queries by their frequency and split them into three quantiles, and denote the queries in the top quantile as the head queries (`HQ`), those in the second quantile as the torso queries (`TQ`), and those in the third quantile as the tail queries (`LQ`). For our experiments, we focus on the top-2 quantiles, i.e., `HQ` and `TQ`, and draw a uniform sample of 250 query prefixes from each.

Query prefixes and suggestions: To simulate a scenario where the user has only typed part of the query and is observing as-you-type query completions, we use the first 30% characters of the queries only. We experimented with other methods for generating query prefixes (e.g., using words instead of characters, or varying the fraction of characters) observing similar results. For each prefix p, we collect the top-15 query suggestions returned by Yahoo! Search.

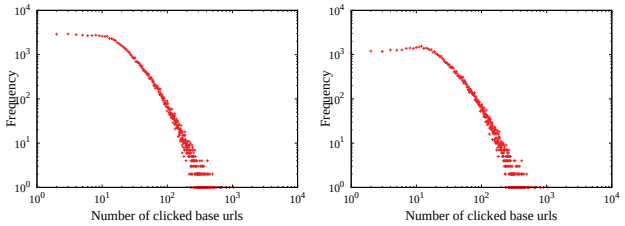

Figure 4: Distribution of number of click urls for prefixes in `HQ` (left) and `TQ` (right).

Clickthrough data: To generate the clickthrough graph (see Section 2), we used fully anonymized query clickthrough logs for first three months of 2009. Figure 4 shows the distribution of the number of clicked results for the two query sets, `HQ` and `TQ`.

User studies: All user studies and manual annotation tasks described in this Section were performed by a group of eight professional search engine quality evaluators experienced with assessing the quality of query suggestions and search results.

5.2 A Pilot Study

Usefulness of query suggestion clusters: To explore whether users would benefit from clustering of search suggestions in the first place, we conducted a small-scale user study. Our annotators were asked to look at two versions of suggestions displayed for 50 different queries: non-clustered (as currently shown on a search engine) and manually clustered (simulating performance of a perfect clustering algorithm). The annotators were asked whether they prefer the clustered on unclustered version; results are shown in Table 1, and although the sample size is relatively small they indicate that clustering has potential value.

Category	Percentage cases(%)
Clustered significantly better	34
Clustered somewhat better	30
No preference	12
Unclustered somewhat better	22
Unclustered significantly better	2

Table 1: User preferences for suggestion clustering in our pilot study comparing unclustered suggestions with clustered suggestions presentation.

Prefixes where clustering was preferred by our annotators included 'ear,' (sample suggestions are, 'ear ache,' 'google earth,' 'earthquake'), 'yos' (sample suggestions are 'yosemite national park,' 'yoshimura,' 'yoshi's') and 'kho' (sample suggestions are, 'hayden kho,' 'khols,' 'khou 11 houston.' Prefixes where clustering was not preferred included 'usps,' where all the suggestions involved the entity 'usps' (e.g., 'usps tracking,' 'usps delivery confirmation,' 'usps rates'), and 'drew,' where a set of unrelated suggestions were presented (e.g., 'nancy drew,' 'drew barrymore,' 'drew university'). In a nutshell, this study confirms that clustering sets of suggestions that correspond to more than one facet (or topic) is desirable, except for the case where the number of facets (or topics) is close or equal to the number of suggestions. As

we will see later, we use these observations when designing our classification and presentation methods.

Cluster prevalence and size: Next, we explore the properties of possible topical clusters in the top suggestions for a query prefix. For this, our annotators constructed a *gold-set* of clusters for all 500 query prefixes. Each annotator was presented with a prefix along with the top-15 suggestions associated with it. Annotators were requested to label suggestions that belong to similar real-life facets subject to two main constraints: (a) there must be at least one cluster with more than a single query suggestion, and (b) participants must try to minimize the number of clusters. Participants were allowed to conduct a research on the web about the query or the suggestions. Overall, this user study resulted in 7500 triplets of the form {*prefix, suggestion, cluster-id*}. Figure 5(a) shows the distribution of cluster sizes over all 500 queries; Figure 5(b) shows the average size of the clusters, when the number of top suggestions being clustered is between 2 and 15 per query. Note that, on average, clusters are relatively small — indicating a high level of topical ambiguity within the suggestions.

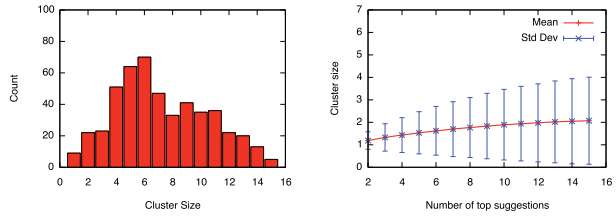

Figure 5: Mean and standard deviation of the cluster size within each top-n query suggestion set.

Usefulness and choice of suggestion cluster labels: Our second pilot study studied whether users would benefit from labeling clusters of suggestions. Our annotators were now shown two versions of 400 suggestion clusters, each containing more than one suggestion each: an unlabeled presentation and one where labels were manually generated. The annotators were asked whether they prefer the unlabeled or labeled version; overall, for 238 (60%) of the clusters a labeled version was preferred – a small, yet consistent gain across different individual annotators.

5.3 Clustering Query Suggestions

After confirming the usefulness of clustering query suggestions for a prefix, we extensively evaluate our techniques for clustering suggestions for a prefix. We discuss our evaluation methodology and metrics before reporting the results.

5.3.1 Experimental Settings

Evaluation methodology: To evaluate the quality of the clusters generated by our methods, we used the clustering gold-set described earlier, containing full manual clusterings of 500 query prefixes.

Techniques to compare: We evaluate our proposed suggestion clustering methods from Section 2. We are unaware of any existing system for providing facet-aware search suggestions for search engine users. However, arguably one natural extension of the current search assistance mechanism is to group suggestions by shared prefix. Specifically, given a prefix p for which we have a set of suggestions S, we split each suggestion s_i into a sequence fr where f contains all the characters in s_i until the first occurrence of p, including p; r is the remainder of the characters. We group together suggestions in S that share first k characters in r, and pick k as 30% of the length of remainder characters, r. We use this shared-prefix method as a baseline; this results in the following methods to cluster suggestions for a prefix.

- `B-pre`: Our strong baseline, using charachter overlap after the occurrence of the prefix in each suggestion.
- `Head`: Clustering method based on indicative words in the suggestions.
- `Search`: Clustering method based on the top-10 results from a search engine.
- `Click`: Clustering method based on clickthrough data from query logs.

Evaluation metrics: To test the performance of each clustering method, we use standard clustering evaluation metrics: two set-based measures, namely *purity* and *inverse purity*, and a pair-based measures, namely, *Rand statistic*, and cluster *entropy* [1]. We define these measures as:

Purity: Purity of a set of clusters is computed by assigning each cluster (C_i) to the most frequently labeled category (L_j) and computing the fraction of correct assignments.

$$\text{Purity(S)} = \frac{1}{|S|}\sum_{C_i \in C} max_j(|C_i \cap L_j|)$$

Inverse purity: While purity measures the precision of a cluster, it can be trivially maximized by generating clusters of size 1. To measure how well similar suggestions were grouped together, we compute the inverse purity defined as:

$$\text{Inverse purity(S)} = \frac{1}{|S|}\sum_{L_j \in L} max_i(|L_j \cap C_i|)$$

F-measure: Similar to the F-measure frequently used in information retrieval, we can combine purity and inverse purity to derive a single metric:

$$\text{F-measure(S)} = \sum_{L_j \in L} \frac{|L_j|}{|S|} max_j\{F(L_j, C_i)\}$$

where $F(L_j, C_i) = \frac{2 \cdot R(L_j,C_i) \cdot P(L_j,C_i)}{R(L_j,C_i)+P(L_j,C_i)}$, $P(L, C) = \frac{|C \cap L|}{|C|}$ and $R(L, C) = \frac{|C \cap L|}{|L|}$

Rand statistic: A clustering method can be viewed as a decision-making process where given a pair of suggestions (s_i, s_j), the method decides whether they should be grouped together or not. The Rand statistic R evaluates the performance of the clustering method for a total of $\frac{|S| \cdot (|S|-1)}{2}$ decisions and is defined as:

$$\text{R(S)} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP is the number of similar suggestions that were assigned to identical clusters, TN is the number of dissimilar suggestions that were assigned to non-identical clusters, FP is the number of similar suggestions that were assigned to non-identical clusters, and FN is the number of dissimilar suggestions that were assigned to identical clusters.

Entropy: Entropy of a set S of clusters is defined as:

$$\text{Entropy(S)} = -\sum_{C_i \in C} \frac{|C_i|}{|S|} \sum_{L_j \in L} Pr\{C_i, L_j\} \times log_2 Pr\{C_i, L_j\} \quad (7)$$

where $Pr\{C_i, L_j\}$ is the probability of finding a suggestion from cluster C_i in labeled category L_j.

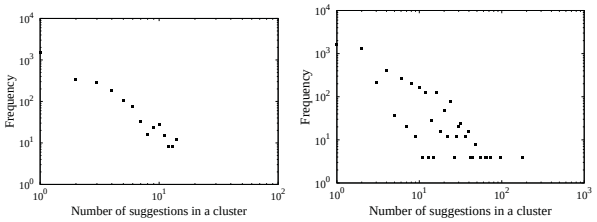

Figure 6: Distribution of number of clusters generated using `Head` (left) and (b) `Click` (right).

5.3.2 *Experimental Results*

Table 2 reports evaluation measures for all methods; statistical significance with respect to `B-pre` is measured using the sign test [19].

Method	Purity	Inverse purity	F-measure	Entropy	*R*-statistic
`B-pre`	0.755	0.841	0.684	0.786	0.734
`Head`					
`first`	0.680†	0.879†	0.646†	0.130†	0.674†
`last`	0.699†	0.783†	0.685†	0.120†	0.682†
`freq`	0.769†	0.857†	0.733†	**0.090**†	0.791
`Search`	0.498	0.944	0.505	0.235	0.437
`Click`	**0.851**†	0.777†	**0.737**†	0.417†	**0.808**†

Table 2: Performance of different clustering methods over all prefixes. († indicates statistical significance over baseline `B-pre`.)

Clearly, clustering suggestions becomes harder as more suggestions are displayed to the user. Figure 7 shows the decay in performance of two of our clustering approaches, `Head` and `Click`, when the number of suggestions being clustered is increased.

To further understand the nature of clusters generated by our clustering methods, we examined the distribution of the number of clusters as well as the size of the clusters generated by each method. Figure 6 shows the distribution of the sizes of clusters for `Click` (Figure 6(b)) and `Head` (Figure 6(a)). In general, `Click` tends to generate more clusters as compared to `Head`. We traced our observations on the evaluation metrics by examining a few examples cases of suggestion clusters. Table 3 shows a sample of clusters generated by three of our methods, namely, `Head` (using `freq`) , `Click`, and `Search`. Examples show how using using `Click` is influenced by the navigational behaviour of users which can, in turn, lead in smaller clusters. For instance, in case of 'atla,' although suggestions, 'atlanta braves' and 'altanta falcons' are associated with the same geographical location, i.e., 'atlanta,' they are placed into two different clusters due to the fact that users interested in these suggestions mostly click on dissimilary URLs. On the other hand, using `Head` generates a single cluster for suggestions related to 'atlanta' due to their high associativity. As a counter example, consider the case where the prefix is 'bp ga.' Here the suggestions 'bp gas mania' and 'bp gas mania game' that correspond identical facets were placed in different clusters. This is due to the fact that the associativity statistics used by `Head` do exhibit a high similarity between the terms 'bp gas' and 'game' than that between 'bp gas' and 'mania.' In contrast, `Click` based on the fact that the suggestions, 'bp gas' and 'bp gas mania' generated a click on similar urls, could correctly cluster these suggestions.

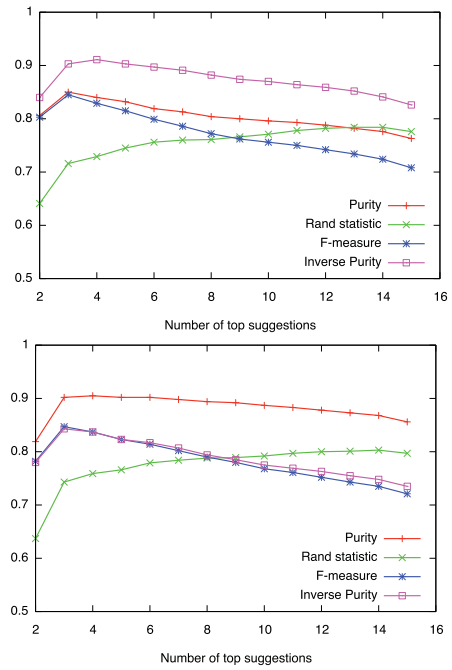

Figure 7: Clustering performance on varying number of suggestions: `Head` (top) and `Click` (bottom).

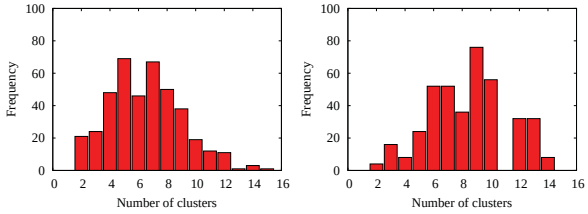

Figure 8: Distribution of number of clusters generated using `Head` (left) and (b) `Click` (right).

In conclusion, we observed that our methods `Click` and `Head` outperform our strong baseline, `B-pre`, for a range of evaluation metrics. We also observed that `Click` and `Head` may substantially differ in the sizes and nature of clusters they produce, with `Click` generating pure but smaller clusters and `Head` generating larger but homogenous clusters. The choice of clustering method naturally depends on the nature of the prefix and the general intent of users that submitted a prefix. Combining the virtues of the two proposed clustering methods in a principled manner remains our future work.

5.4 Labeling Suggestion Clusters

Evaluation methodology: Finally, we evaluate the quality of the labels generated for the clusters. For this, we used all of the proposed labeling methods to assign labels to the clusters of suggestions generated for 250 query prefixes used for the previous experiments; overall, the set contained 1039 clusters. Our annotators judged each label as correct on incorrect; a label was deemed correct for a cluster if it describes the suggestions in it in a way that assists a user in understanding why these suggestions appear together. La-

Head	Search	Click
dji	dji	dji
dji index	dji index	dji index
djia chart	djia chart	djia chart
djia futures	djia futures	djia futures
djia stocks	djia stocks	djia stocks
————	————	————
djibouti	djibouti	djibouti
djibouti africa	djibouti africa	djibouti africa
————	djibouti africa map	djibouti africa map
djibouti africa map		
bp gas	bp gas	bp gas
bp garage	————	————
bp garage locations	bp garage	bp garage
bp gas station locations	————	————
bp gas credit cards	bp garage locations	bp garage locations
————	————	bp gas credit cards
bp gas mania games	bp gas credit cards	bp gas mania game
bp gas mania	bp gas mania game	bp game
bp game	bp game	bp gas station locations
————	bp gas station locations	————
	————	bp gas mania
	bp gas mania	
atlantis	atlantis	atlantis
atlantis bahamas	atlantis bahamas	atlantis bahamas
atlantis resort	atlantis resort	atlantis resort
————	————	————
atlanta braves	atlanta braves	atlanta braves
————	atlanta falcons	atlanta falcons
atlanta falcons	————	atlanta airport
————	atlanta airport	craigslist atlanta
craigslist atlanta	craigslist atlanta	
————		
atlanta airport		

Table 3: Sample suggestion clusters generated by various methods for the query prefixes *dji* (top), *bp ga* (middle), *atla* (bottom).

bels with incorrect spellings, incomplete words, or incoherent words are considered as incorrect ones.

Techniques for comparison: We compare the different approaches described in Section 3:

- `MFS`: Our strong baseline, using overlapping characters after the occurrence of the prefix in each suggestion.
- `LCS`: Clustering method based on indicative words in the suggestions.
- `MFRS`: Most frequent word n-gram in the result set, where the suggestions are used as queries.
- `MFRS*`: Most frequent word n-gram in the result set, where the words of the suggestions that are not shared are used as queries.
- `Comb`: The combined labeling approach.

Evaluation metrics: To test the accuracy of the labeling we measure the precision of the method – the percent of correct labels assigned by it. Since in many cases the correct label is assigned by more than one method, we also track, for each method, the fraction of clusters for which the correct label was produced by this method only. We refer to this latter metric as the *lead* of the method.

Table 4 shows the precision of the different labeling approaches we tested. Examining the labeling errors, we observed that most stem from imperfect clustering results: the less coheseive a cluster is, the less likely it is that techniques like LCS or MFRS will result in a meaningful cluster.

Method	Precision	Lead
MFS	0.642	0.064
LCS	0.611	0.103
MFRS	0.694	0.178
MFRS*	0.493	0.034
Comb	**0.719**	*n/a*

Table 4: Performance of cluster labeling methods.

5.5 Ranking Suggestion Clusters

Earlier in Section 4 we proved the correctness of our algorithm to rank suggestion clusters at minimizing the user effort when formulating a search query. We put our cost metric to test by examining whether our clustering algorithm places desirable suggestion clusters at the top. In particular, we recruited human annotators and presented them with a set of suggestion clusters. Participants were requested to pick rank two suggestion clusters that they would like to see at rank 1 and 2. It is noteworthy that this is a stricter evaluation than presenting human annotators with a list of ordered clusters and requesting whether they agree or not with the presented ranking.

For a set of 50 prefixes, we observed that for 85% of the prefixes the cluster placed at rank 1 by our ranking algorithm was also picked as the top-1 cluster by the human annotators. Additionally, for 13% of the prefixes our ranking algorithm placed the top-1 manually chosen cluster at rank 2. Interestingly, 2 of such cases were where the beginning of the prefix did not overlap with the suggestions in the first cluster. For instance, for the prefix, 'pot,' two competing clusters were {'harry potter, 'harry potter movie', 'harry potter and the half-blood prince'} and {'potato','potato soup'}. While our ranking algorithm ranked the first cluster higher than the second, our human annotators ranked the second cluster higher. The explanation for their choice as traced by comments was: '*A searcher would type harry pot if they were interested in harry potter.*' We believe that this is a special case of query completion, i.e., not all auto-completion allow for non-overlapping prefix of each suggestion in a set, and only a relatively small set of prefixes follow this trend. Our cost-metric can be easily extend to handle such cases.

5.6 Evaluation conclusion

In summary, both `Click` and `Head` outperform the baseline methods for clustering suggestions, with `Click` exhibiting higher values for f-measure and `Head` generating clusters with lower entropy. Interestingly, clustering based on terms in search results (`Search`)—an approach shown to be effective in general query clustering—does not perform as well when applied to search suggestions. On the other hand, using the search results to obtain cluster labels (`MFRS`) produces high-precision labels while outperforming other methods; combining these labels with labels found in the suggestions themselves improves accuracy futher. Finally, we verified that our cluster ranking algorithm generates an desirable ordering that meets the user needs.

6. RELATED WORK

Suggestions for query completions: Predictive text generation systems, offering possible completions to the user's text based on the existing input, have been used since the 1970s, although their applications have mostly been restricted to assisting people with physical disabilities [13]. In recent years, these systems have been found to be beneficial to broader audiences in mobile devices [11] and in web search engine interfaces. In particular, in the case of search engines, the user is often not aware of the wording used in the web pages she is searching for, making the suggestive framework useful not only for predicting what the user is likely to type, but also for offering query forms the user does not necessarily consider, and that lead to relevant information.

Today, all major search engines offer search suggestion technologies, and similar technologies adapted in related areas such as databases [17]. However, although much work exists on deriving related queries [16, 28, 3, 20], little research has focused on organizing—and in particular, clustering—query suggestions. Boldi et al. describe an approach for labeling query reformulation types [4]; while this can be used for organizing query suggestions, the labels attached to the reformulations are functional in nature (e.g. "generalization") rather than topical as in our approach.

Query similarity and clustering: Research on clustering of web queries is mostly concerned with large-scale clustering – that is, grouping an entire query log into different topics, often for the purpose of query recommendation or for analysis of topics of interest over a large population. As with most clustering tasks, the core issue addressed is estimating the distance between the items to be clustered: in this case, queries. Since these tend to be short, lexical distance measures perform poorly, and the approaches developed for the task of query similarity estimation typically use information external to the query. Glance measures similarity using the overlap between the sets of retrieved documents for queries [15]; Sahami and Heilman use the terms appearing in documents retrieved for a query to compute the similarity between queries [23]. Similarly, Beeferman and Berger [2] and Wen et al. [26] use the set of clicked documents for a given query to determine its cluster. Chien and Immorlica measure the distance between queries according to the similarity between their temporal profile [8]. Closest to our work, is that of Sadikov et al. [22] which proposes a mechanism for grouping *post-submit* query suggestions by performing random walks on the query reformulation graph in an offline step; however, to the best of our knowledge, there has been no published work on clustering *pre-submit* suggestions.

Labeling clusters: In the context of web search, work on labeling clusters is mostly focused on clusters of documents [12, 24, 7]. In this setting, labels are often chosen from titles of central documents, frequent terms appearing in the documents of a cluster, or recurring named entities. Targeting shorter texts, Pantel and Ravichandran propose an approach to labeling sets of concepts where representative elements of the set, appearing in one of several known grammatical structures that indicate a label, are used to identify and rank candidates for a cluster label [21]. A similar approach is described by Chung et al. [10], using Wordnet to expand signatures defined for each cluster of concepts. These strategies work well when the instances of the clusters to be labeled are concepts sharing the same semantic type, and when the cluster size is relatively large (so that representative elements can be mined. In the case of query suggestions, the clusters to label are small in size, and often contain mixed semantic classes; in fact, in many cases, a suggestion will contain more than a single concept, or no concepts at all.

7. CONCLUSIONS

Query auto-completion increases the usability of a web search engine substantially; despite this, the manner in which queries are completed has remained unchanged since the introduction of this feature on major web search engines. In this paper, we propose an alternative presentation of as-you-type query suggestions, one where—for some queries—suggestions are grouped by topic. We show that users prefer this suggestion mechanism over an unclustered presentation, and evaluate several approaches to performing the clustering, with an increasing level of knowledge available to a search engine. We also discuss several additional tasks related to the clustering: selection of the queries to cluster, assignment of a label to each cluster, ordering the clusters themselves and the suggestions within each cluster. We accompany each of the tasks we address with rigorous evaluation using real-life data collected from a major search engine.

To facilitate further research into alternative presentations of search suggestions in general and their clustering in particular, we plan to make our data publicly available.

8. REFERENCES

[1] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4):461–486, 2009.

[2] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD*, 2000.

[3] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and Sebastiano. Query suggestions using query-flow graphs. In *WSCD 09*, 2009.

[4] P. Boldi, F. Bonchi, C. Castillo, and S. Vigna. From 'dango' to 'japanese cakes': Query Reformulation Models and Patterns. In *WI*, 2009.

[5] D. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *WWW*, 2007.

[6] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 2006.

[7] D. Carmel, H. Roitman, and N. Zwerdling. Enhancing cluster labeling using wikipedia. In *SIGIR 2009*, 2009.

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

[9] S.-L. Chuang and L.-F. Chien. A practical web-based approach to generating topic hierarchy for text segments. In *CIKM*, 2004.

[10] C. Y. Chung, R. Lieu, J. Liu, A. Luk, J. Mao, and P. Raghavan. Thematic mapping - from unstructured documents to taxonomies. In *CIKM 2002*, 2002.

[11] K. Church and B. Thiesson. The wild thing! In *ACL*, 2005.

[12] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *SIGIR'92*, pages 318–329, 1992.

[13] J. J. Darragh and I. H. Witten. *The Reactive Keyboard*. Cambridge University Press, New York, NY, USA, 1992.

[14] S. Dumais, E. Cutrell, and H. Chen. Optimizing search by showing results in context. In *CHI*, 2001.

[15] N. Glance. Community search assistant. In *IUI*, 2001.

[16] R. Jones, B. Rey, O. Madni, and W. Greiner. Generating query substitutions. In *WWW*, 2006.

[17] G. Li, S. Ji, C. Li, and J. Feng. Efficient type-ahead search on relational data: a tastier approach. In *SIGMOD*, 2009.

[18] D. Lin. Automatic retrieval and clustering of similar words. In *ACL*, 1998.

[19] J. P. Marques De Sá. *Applied Statistics*. Springer Verlag, 2003.

[20] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM*, 2008.

[21] P. Pantel and D. Ravichandran. Automatically labeling semantic classes. In *HLT-NAACL*, 2004.

[22] E. Sadikov, J. Madhavan, L. Want, and A. Halevy. Clustering query refinements by user intent. In *WWW*, 2010.

[23] M. Sahami and T. Heilman. A Web-based kernel function for measuring the similarity of short text snippets. In *WWW*, 2006.

[24] P. Treeratpituk and J. Callan. Automatically labeling hierarchical clusters. In *DG 06*, 2006.

[25] P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *ECML*, 2001.

[26] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Query clustering using user logs. *ACM Transactions on Information Systems*, 2002.

[27] W.-T. Yih and C. Meek. Improving similarity measures for short segments of text. In *IAAI*, 2007.

[28] Z. Zhang and O. Nasraoui. Mining search engine query logs for query recommendation. In *WWW*, 2006.