Learning to Suggest: A Machine Learning Framework for Ranking Query Suggestions

Umut Ozertem[*]
Speech @ Microsoft
Mountain View, CA
umuto@microsoft.com

Olivier Chapelle[*]
Criteo
Palo Alto, CA
o.chapelle@criteo.com

Pinar Donmez[*]
Salesforce
San Francisco, CA
pdonmez@salesforce.com

Emre Velipasaoglu
Yahoo! Labs
Sunnyvale, CA
emre_velipasaoglu@yahoo.com

ABSTRACT

We consider the task of suggesting related queries to users after they issue their initial query to a web search engine. We propose a machine learning approach to learn the probability that a user may find a follow-up query both useful and relevant, given his initial query. Our approach is based on a machine learning model which enables us to generalize to queries that have never occurred in the logs as well. The model is trained on co-occurrences mined from the search logs, with novel utility and relevance models, and the machine learning step is done without any labeled data by human judges. The learning step allows us to generalize from the past observations and generate query suggestions that are beyond the past co-occurred queries. This brings significant gains in coverage while yielding modest gains in relevance. Both offline (based on human judges) and online (based on millions of user interactions) evaluations demonstrate that our approach significantly outperforms strong baselines.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Query Formulation, Retrieval Models

Keywords

search assistance, query suggestion, machine learning, query log mining

1. INTRODUCTION

Query suggestions are an integral part of the modern search experience. Here we are concerned with query reformulation suggestions that are presented the users after they submit their query, rather than query completion suggestions. The latter is useful in helping the user formulate an original query by aiding in spelling and term selection while typing in the query. Query reformulation suggestions, on the other hand, are useful for changing direction once a search query has been already issued. The needs for such an action could include disambiguation once the user realizes after inspection of the results that the concept he had in mind had possibly other stronger meanings different than his. Or, the user may want to formulate an even more precise and succinct query to hone in on better answers. It could also involve modification of the information need based on the information gained by inspection of the search result, to either dive deeper into research or move to a different aspect of a task. The needs are much varied.

Past scientific research is focused on a few main methods to address most user needs for query reformulation. A large group of methods are based on leveraging the "wisdom of crowds" by analyzing the search logs. Another group of methods use term semantics to derive new queries from existing ones. The former group of methods can be divided into those that exploit query co-occurrences in the search logs, and those that leverage the document click information such as random walks over query-document bipartite graphs. In the latter group, a number of query synthesis methods exist, either synthesizing new queries with active user participation, or directly without any user input. We delve into a review of these existing methods in the related works section, and contrast them with our method.

There are a few fundamental shortcomings of these methods, (i) the utility of a query reformulation suggestion to the user's search task is indirectly handled (ii) all co-occurrences in the query logs are treated equally, without modeling the probability that the pair of queries belong to the same search task or not (iii) models that are solely based on collocated queries in the past logs will have limited coverage.

In this paper, we develop a machine learning framework which addresses the above shortcomings. First, the utility argument is handled explicitly by building a utility model that takes into account positions of URLs that are common to result sets of the original query and the suggestions. Second, we propose an implicit task boundary method to model whether a following query is a *continuation* of the preceding queries; as a result, co-occurrences in the search logs are not

[*] This work was completed while the author was at Yahoo! Labs.

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.
Copyright 2012 ACM 978-1-4503-1472-5/12/08 ...$10.00.

treated equally, but weighted with the probability that they belong to the same task. Finally, we learn to predict the utility of suggestion by using a rich feature space including lexical and result set features that capture the salient factors of why a suggestion could be useful to the user given his initial query, rather than only relying on the observed query co-occurrences in the past. We train this machine learning model with a learning target derived from the past query co-occurrences rather than relying on human judgments. This allows us to generalize the knowledge trained on the past query co-occurrences and the machine learned model can generate both relevant and useful (query,suggestion) pairs that have never observed in the past logs, leading to a significant increase in coverage. Also, it avoids the costly and time consuming human labeling process. Furthermore, the machine learning framework also has the advantage of providing the apparatus that can effectively blend disparate sources of query suggestion candidates such as those based on wisdom of crowds and those based on synthesis, which has been addressed in an ad-hoc fashion at best so far.

In the next section, we review the existing literature and contrast our method with related works. In Section 3, we describe the proposed framework in more detail. Section 4 dives into the description of how we address utility explicitly by constructing a suitable learning target. Section 5 describes the feature space, the training set and the learning method. Section 6 talks about one strength of this method which can score and blend in synthetic suggestion candidates as well as query extensions observed in the search logs that are too rare for the collocation based methods to capture. These are followed by experimental results in section 7 and conclusions in section 8.

2. RELATED WORK

Leveraging "wisdom of crowds" has been very popular in generating query suggestions. A large set of research articles focused on leveraging the session structure and other information to find alternate queries to suggest. The main idea is to find pairs of queries that frequently co-occur in users' search sessions and use them as suggestions for each other. For example, Huang et al. [10] find such co-occurrences and rank the suggestions for an input query by frequency of co-occurrence, $freq(q_1, q_2)$, where q_1 is the input query and q_2 is a suggestion candidate (within a certain time window in the same user session). Jensen et al. [12] considered *point-wise mutual information* (PMI) along with frequency of co-occurrence. PMI is defined as

$$\textsc{pmi}(q_2, q_1) = \log\left(\frac{\Pr(q_2, q_1)}{\Pr(q_1)\Pr(q_2)}\right) \tag{1}$$

For a particular q_1, the $\Pr(q_1)$ term is constant for all the possible suggestion candidates, and ordering given by PMI only depends on the ratio of the frequency of q_2 conditioned on q_1 divided by the marginal frequency of q_2. While, frequency of co-occurrence is not a test of dependence, PMI is a valid test of independence. However, PMI favors rare events [24]. A preferred test of collocations in text corpora has been G^2 *log-likelihood-ratio* (LLR), introduced to NLP by Dunning [7]. Moore gives several formulations of G^2 in [17] and shows it is equivalent to *mutual information* (MI)

$$\begin{aligned}\textsc{mi}(q_2, q_1) = {} & \Pr(q_2, q_1)\textsc{pmi}(q_2, q_1) + \Pr(q_2, \overline{q}_1)\textsc{pmi}(q_2, \overline{q}_1) \\ & + \Pr(\overline{q}_2, q_1)\textsc{pmi}(\overline{q}_2, q_1) + \Pr(\overline{q}_2, \overline{q}_1)\textsc{pmi}(\overline{q}_2, \overline{q}_1)\end{aligned} \tag{2}$$

where $\overline{q}$ denotes the set of all queries except q. MI addresses the bias of PMI by taking into account the probabilities of the complement events as well. LLR was found to be among the best performing statistical tests of dependence by Thanopoulos et al. [24] for extracting collocations. LLR was used by Jones et. al. [13] as a feature in a machine learning framework to expand queries for matching advertisements to an input query.

Boldi et al. [3] move to a more structured processing of the sessions by building a query-flow graph where the nodes are queries and edges are associated with weights that capture how likely a user is to move from one query to the next within a session. Then, neighbors with the largest edge weights are selected as suggestions for an input query. In [2], they also classified the transitions as specializations which are essentially query extensions, generalizations which are usually contractions, errors corrections and lateral moves, and use these categories for selecting suggestions for different purposes.

Another group of methods focused on leveraging the clicked documents by building a query-document bipartite graph. Assumption here is that similar queries have larger overlap between their respective clicks. For example, Mei et al. [15] use a random walk over the bipartite graph to find similar queries. Beeferman et al. [1] use hierarchical agglomerative clustering iteratively to find groups of queries that are similar and can be used as suggestions for one another, however, this is expensive. Baeza-Yates et al. [21] used an efficient k-means algorithm to find similar queries, but it requires the number clusters ahead of time. Sadikov et al. [22] combined the query-flow graphs with document click information to find query suggestions.

Query synthesis methods looked into generating suggestions by leveraging search logs as well as external information sources. Szpektor et al. [23] use a template generation method by leveraging WordNet [16]. Use of all token boundaries in segmentation of the queries leads to many poor suggestion candidates. Jain et al. [11] use Conditional Random Fields to segment to queries, yielding better results. They also use a machine learned stage to filter bad suggestions but they do not address blending the suggestions with other methods such as those based on session analysis. Both in [11] and [13] the machine learning for suggestion scoring is based on editorial labels, whereas in our method we use probabilities estimated from the query logs as regression target.

3. PROPOSED FRAMEWORK

This section gives an overview of our proposed pipeline. The detailed description of each component follows in the subsequent sections. Below is a brief summary of each component.

- **Target Generation**: The first step deals with estimating a scoring function that measures how useful and relevant is a follow-up query to a given query. We estimate this score by a probabilistic utility function that relies on the query co-occurrence. We discuss different definitions of the query co-occurrence in the next section. The scores are used as the target values in our machine learning model. This is a discriminating advantage of our method and one of our main contributions since it saves the costly and time consuming human labeling process.

- **Features**: The next step of the pipeline is to create the feature vector representation of each query pair in the data. We use two sets of features, namely lexical and result set based features. They are designed to capture the semantic similarity and helpfulness of the suggestion to the original query which is formally defined in the following section. Since the features do not depend on the session logs, it allows our system to handle synthetically generated suggestion candidates. This is another differentiator of our approach when learning query suggestions.

- **Ranking Model**: Once the target scores are estimated and the features are generated, we perform regression using Gradient Boosted Decision Tree (GBDT). This model enables us to rank the suggestion candidates for a given a query, and eliminate the irrelevant and useless ones.

- **Candidates**: The candidate suggestions we tested the ranking model on come from various sources including query logs. However, query logs provide only limited number of candidates especially for rare queries. To address the sparsity, we use additional candidate queries that are either synthetically generated or extended queries. Our feature space is appropriate to blend in such candidates with those obtained from query logs.

4. TARGET GENERATION

4.1 Query Co-occurrence

The targets to be used in our machine learning model depend on $\Pr(q_2, q_1)$, namely the probability of *query co-occurrence*. In this paper, we define query co-occurence as when two queries are manually[1] issued by the same user within the same *session*. We define the user session as all the user activity within a time window limited by 10 minutes of inactivity. Nevertheless, the models we develop in this paper do not strictly depend on this definition and can be used with other timeout limits or other definitions of user session.

A first design choice is whether to consider all pairs of queries within the same session or consecutive queries only. The latter option makes the query co-occurrence set robust to intent drift in the search session. The intent drift can be defined as the gradual change in user's intent as the search session progresses, and perhaps can be explained best by an example. Here is a real search session: "lost cast" → "dexter cast" → "michael c. hall" → "michael c. hall cancer". The argument against restricting to consecutive pairs is that the majority of sessions do not have such intent drift and it results in a significant decrease in coverage. We will provide experiments with both of these choices.

The simplest approach for scoring suggestion candidates is to measure the reformulation probabilities from the frequency counts in the logs and pick the queries with the highest reformulation probabilities $\Pr(q_2 \mid q_1)$,

$$\Pr(q_2 \mid q_1) = \frac{\Pr(q_2, q_1)}{\Pr(q_1)} \tag{3}$$

Here $\Pr(q_2, q_1)$ is the probability that q_2 occurs after q_1 within the same session and $\Pr(q_1)$ is the marginal probability of q_1. There are two problems with this approach. First it could be that the query q_2 is unrelated to q_1. This happens when q_1 and q_2 belong to different *tasks* that the user wants to solve. One way to address that issue would be to consider only queries within the same task, but this requires a system for detecting task boundaries [2] which itself can be prone to errors. In this paper we propose a solution that does not rely on task boundary detection. The second problem is that the result page associated with query q_2 may not be useful, for example if the documents that q_2 retrieves are identical of those of the original query q_1.

For these reasons, we say that, given a query pair (q_1, q_2), the query q_2 was a *helpful* reformulation of query q_1 if and only if the following two conditions are satisfied:

1. The query q_2 is a *continuation* of q_1. If the query q_2 is the beginning of a new task and has nothing to do with q_1, we should not consider q_2 to be a helpful reformulation for q_1.

2. The query q_2 has a positive *utility*, that is the search results returned for that query are useful to the user.

The details of these two conditions are described in the next two sections.

4.2 Utility of Reformulations

When trying to assess the utility of a reformulation, a simple criterion is to say that reformulations that lead to a click in the result page of q_2 are useful, and the others are not useful. In the extreme case, a query for which the search engine does not return any results cannot be a good suggestion by definition.

But reformulations followed by a click are not always useful. Consider the following query reformulations that appear frequently in query logs "bank of america" → "bank of america online" or "facebook" → "facebook login". Although co-occurrences like these lead to a click on the result set of the second query q_2, they do not likely take the user to a destination URL that is not already directly accessible from the original query q_1. We thus define a reformulation to be useful only if it leads to a click on a URL that either is not existing in the search result page of q_1 or that is ranked higher than that in the search result page of q_1.

For formalizing this idea we compare the ranks of the same URLs in q_1 and q_2 (if any) and use the rank discounts that DCG uses. Let D_c be the set of clicked documents on the result page of q_2 and $r(q, d)$ returns the rank of the given document $d \in D_c$ for the given query q and returns $+\inf$ if the URL is not ranked. The total difference in discounts of the clicked documents Δ is defined as

$$\Delta = \sum_{d \in D_c} \left(\frac{1}{\log r(q_2, d)} - \frac{1}{\log r(q_1, d)} \right) \tag{4}$$

When there was a click on q_2, the reformulation is defined to be useful if $\Delta > 0$. This occurs only if at least one of the two conditions above holds for one of the clicked url.

In addition to the condition in (4), we also consider the queries that can address the user need directly in the search result page (without any clicks) as useful [5]. For example, weather information in the weather direct answer module for the query "lake tahoe weather", or current stock quote

[1] We ignore queries suggested by the system because they may introduce a *presentation bias*.

Table 1: Effect of the implicit session model (see section 4.3).

$\Pr(q_2 \mid q_1)$		$\Pr(q_2 \mid q_1, c=1)$	
facebook	0.0286	rei	0.0431
ebay	0.0244	nordstrom	0.0342
amazon	0.0225	dicks	0.0307
rei	0.0214	columbia	0.0281
nordstrom	0.0191	macys	0.0274

q_1 = northface

$\Pr(q_2 \mid q_1)$		$\Pr(q_2 \mid q_1, c=1)$	
facebook	0.0794	chase	0.0993
chase	0.0415	capital one	0.0558
wells fargo	0.0227	wells fargo	0.0530
capital one	0.0223	american express	0.0404
google	0.0185	bank of america online	0.0281

q_1 = bank of america

and other details in the finance module for the query "amzn" are examples of such queries, where the user does not need any clicks to get to the desired information. Therefore in addition to the condition in (4), if there is no further query reformulation in the session, and if the final query of the session contains such direct answer modules, we consider these queries to be useful reformulations as well.

To sum up, we define a query reformulation to be useful if either there was at least a click on q_2 and $\Delta > 0$; or there was no click on q_2, the result page of q_2 contained a direct answer module and there was no further reformulation afterwards.

4.3 Implicit Task Boundary Detection

Remember that we want to compute the probability that q_2 follows q_1 under the condition that q_2 is a continuation of q_1 (denoted by $c = 1$) and that it is useful (denoted by $u = 1$) as defined in the previous section; that is,

$$\Pr(q_2 \mid q_1, c = 1, u = 1). \tag{5}$$

First, note that u is an observed variable, while c is not. It is in particular straightforward to compute $\Pr(q_2 \mid q_1, u = 1)$ by simply filtering out the pairs for which $u = 0$. In the rest of this section, we will thus assume that all probabilities are implicitly conditioned on $u = 1$ and will drop that conditioning from the equations.

The following equation holds:

$$\begin{aligned}\Pr(q_2 \mid q_1) = \Pr(q_2 \mid q_1, c = 0) \Pr(c = 0 \mid q_1)\\ + \Pr(q_2 \mid q_1, c = 1) \Pr(c = 1 \mid q_1).\end{aligned} \tag{6}$$

Let us denote $\mu = \Pr(c = 0 \mid q_1)$. In the case that the user issues an unrelated query ($c = 0$), q_1 and q_2 are independent and thus $\Pr(q_2 \mid q_1, c = 0) = \Pr(q_2 \mid c = 0)$ that we approximate as the marginal distribution $\Pr(q_2)$. Equation (6) becomes:

$$\Pr(q_2 \mid q_1) = \mu \Pr(q_2) + (1 - \mu) \Pr(q_2 \mid q_1, c = 1). \tag{7}$$

In equation (7), both $\Pr(q_2 \mid q_1)$ and $\Pr(q_2)$ are known distributions and the distribution that we want to compute is $\Pr(q_2 \mid q_1, c = 1)$. At a high level, this can be achieved by subtracting $\mu \Pr(q_2)$ from $\Pr(q_2 \mid q_1)$. But we cannot do this subtraction naively as this could lead to negative probabilities. The correct way of computing $\Pr(q_2 \mid q_1, c = 1)$ is a maximum likelihood estimation as explained below.

Let us consider for now that μ is known and fixed. For a given query q_1, let us denote the n unique queries q_2 that cooccured with q_1 as $q^1, \dots, q^n$ and let N_{q^i} be the number of co-occurences. With these notations, we have that $\Pr(q_2 = q^i \mid q_1) \propto N_{q^i}$. Finally, let $p_i = \Pr(q_2 = q^i \mid q_1, c = 1)$ be the probabilities that we want to evaluate. The maximum likelihood estimate can be formulated as the solution of the following optimization problem:

$$\max_{p_i} \sum_{i=1}^{n} N_{q^i} \log[\ \mu\ \Pr(q_2 = q^i) + (1 - \mu)\ p_i\ ], \tag{8}$$

under constraints,

$$\sum_i p_i = 1, \quad p_i \geq 0.$$

In practice, in order to avoid solving a constrained optimization problem, we perform the following change of variables,

$$p_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)},$$

and we are left with a convex and unconstrained optimization problem on x_i which can be solved using standard optimization techniques such as non-linear conjugate gradient descent.

Note that in the extreme case $\mu = 0$, the optimal solution of (8) is $p_i = N_{q^i} / \sum_j N_{q^j} = \Pr(q_2 = q^i \mid q_1)$, which concurs with equation (7).

The only part left is how to select, for each q_1, $\mu = \Pr(c = 0 \mid q_1)$. We use the following intuition: $\Pr(q_2)$ has a very large entropy, but the entropy of $\Pr(q_2 \mid q_1, c = 1)$ should be much smaller because there are only a limited number of queries that users are likely to type as a continuation of q_1. Our heuristic to select μ is to find the value such that the resulting distribution p_i has minimum entropy. On top of this entropy criterion, we add a Beta(1,10) prior on μ, reflecting that for most queries q_1, the following query q_2 is unrelated (large μ).

The importance of conditioning on $c = 1$ is illustrated in table 1. Without this conditioning, irrelevant but frequent queries such as "facebook" tend to have a high $\Pr(q_2 \mid q_1)$ value.

5. MACHINE LEARNING

This section first discusses the feature space that the ranking model uses and then the training process.

5.1 Features

We use two main categories of features: lexical features and result set features as shown in Tables 2 and 3. Most of the features are intuitively simple and the short descriptions are enough, and the rest is detailed here. We do not use session log features because they are not defined for synthetic suggestions as well as other candidate suggestions.

The lexical features use the query strings themselves, like number of words in each query, number of words or characters they share in the beginning or at the end, and Levenshtein distance [14] between the two queries.

In the result set features, there are mainly two types: those that consider the quality of the results and those that

Table 2: Lexical features

Feature	Short Description
LEV	Levenshtein distance between q_1 and q_2
LQ1	length of q_1
LQ2	length of q_2
LDIFF	LQ1-LQ2
ABSLDIFF	abs(LDIFF)
ABSLDIFFN	ABSLDIFF/LQ1
NW1	number of tokens in q_1
NW2	number of tokens in q_2
COMMONW	number of tokens in common in q_1 and q_2
COMMONWN	COMMONW/NW1
COMMONWP	tokens in common in the beginning
COMMONWS	tokens in common at the end
COMMONCP	characters in common in the beginning
COMMONCS	characters in common at the end

Table 3: Result set features

Feature	Short Description
LTR11	LTR score of the top result for q_1
LTR21	LTR score of the top result for q_2
LTR15	average LTR score at top 5 for q_1
LTR25	average LTR score at top 5 for q_2
LTR110	average LTR score at top 10 for q_1
LTR210	average LTR score at top 10 for q_2
LTRRATIO1	LTR11/LTR21
LTRDIFF1	LTR11-LTR21
LTRRATIO5	LTR15/LTR25
LTRDIFF5	LTR15-LTR25
LTRRATIO10	LTR110/LTR210
LTRDIFF10	LTR110-LTR210
COURLCOUNT1	boolean, URLs in top result are the same
COURLCOUNT5	number of the same URLs in top 5
COURLCOUNT10	number of the same URLs in top 10
CODOMAINCOUNT1	boolean, domains of the top results are the same
CODOMAINCOUNT5	number of same domains in top 5
CODOMAINCOUNT10	number of same domains in top 10
CODOMAINIQF1	IQF of of the domain at top result, if the same
CODOMAINIQF5	total IQF of the common domains in top 5
CODOMAINIQF10	total IQF of the common domains in top 10
ABOUTNESS	cosine similarity over the aboutness vectors

consider the overlap of the two result sets. For assessing the quality of the results we use scores given by a learning to rank system (LTR), but any other relevance score such as BM25 could have been used. Queries that have better results should have higher average LTR scores. We use the LTR scores themselves, as well as the difference and the ratio, intuitively, a high quality query suggestion should have good results, even better if it has better results than the original query, which creates assisted paths to relevant and well formed queries.

Another group of features is the number of common URLs and domains in top K ranks. These aim to determine the overlap in the result sets of q_1 and q_2, which has been used as a query-pair level relevance metric [20, 8]. We typically expect the predictions to be high when the values of these features are in a medium range: when the overlap between queries is too small, they are probably not related; and when it is too large, the query q_2 does not contain any new information relative to q_1 (see section 4.2). Number of common URLs gives a strong signal but it is non-zero for only a very small portion of query pairs. To overcome this sparsity issue, we also use the number of common domains. One clear observation is that having results from the same domain is much more informative if the domain is a tail one. As the domain becomes more generic such as `wikipedia.org` or `cnn.com` having such a domain in common becomes less meaningful. To capture this intuition, we use inverse query frequency (IQF) [6], similar to the inverse document frequency well-known in TF-IDF. IQF of a given domain d is

$$\mathrm{IQF}(d) = \log\left(\frac{|Q|}{|q : d \in q|}\right) \tag{9}$$

In words, this is the log of the ratio of number of all queries to the number of queries that lead to at least one result with the given domain d. We generate another set of features using the sum of IQF of the common domains in top K ranks.

Result set aboutness: Aboutness vector similarity is by far the most important (and also the most complicated) result set feature that we use and merits special attention. The problem with the result set overlap based measures is that although they are effective at identifying queries that are very close in meaning (almost synonymous), the overlap drops very sharply to zero when the compared queries are relevant, but not almost identical in meaning. In the context of query suggestions, it is important to identify relevant queries not only almost identical ones. `CODOMAINIQF` solves this problem up to some extent. Yet, one needs compare not only whether the pair of queries return the same results, but also whether the two queries have results that are not identical but are about the same or similar concept. This will give an idea on the semantic relation between a pair of queries.

For example, “toyota prius” and “toyota yaris” are quite related and would be considered good suggestions although have no results in common in top 10, based on the results returned by a web search engine. Hence, the result set overlap is insufficient to assess relevance. This particular one is an example that `CODOMAINIQF` cannot also handle, because although the queries are quite relevant, the common domains that they have are quite popular domains with low IQF scores (`toyota.com`, `autos.yahoo.com`, `autos.msn.com`, `wikipedia.org`).

To build a semantic similarity metric, we construct an aboutness vector of each query, which can be considered as a bag-of-words representation based on the web results, which consists of the *concepts*—that are part of a predefined concept dictionary—in the documents returned for this query. This is based on previous work [19] which computed for each concept and each document an aboutness score defined as the probability that a user interested in this concept would find the document relevant. Algorithm 1 explains how to extend the computation of this aboutness score from documents to query. The score for a query is a weighted sum of the scores of the documents returned for that query, where the weights put more emphasis on documents ranked higher. Finally the aboutness similarity of a query pair is the cosine similarity between the corresponding aboutness vectors. Table 4 shows examples of queries, and their top 20 concept terms ranked in descending order with respect to $S(t)$, the aboutness score of the concept term t. Although quite related, the pairs of queries presented here have zero results in common.

Algorithm 1 Algorithm to compute the aboutness vector

```
Require: Concept dictionary $D$, query $q$.
Retrieve set $R$ of top-$k$ results for $q$.
for concept $t \in D$ do
    $S(t) = 0$
    for $i = 1, .., k$ do
        $d$ = $i$-th document in $R$.
        if concept $t$ is in $d$ then
            $a$ = aboutness score of concept $t$ in $D$ [19].
            $S(t) = S(t) + 0.9^{i-1} a$.
        end if
    end for
end for
Set $S(t)$ to 0 for the concepts $t$ that are not in the top 20
  highest scores.
Aboutness vector: $\mathbf{a}(q) := [\dots, S(t), \dots]$
```

Table 4: Examples of the aboutness vector

Query	Terms in the aboutness vector (ordered)
iphone 4	iphone, store, 4, camera, phone, apps, inc, facetime, mode, recording, software, 3gs, resolution, battery, shop, network, ios, 3g, broadband, ipod
blackberry storm	blackberry, smartphone, verizon, touchscreen, 3g, storm, rim, blackberry, blackberry curve, iphone, vodafone, blackberry pearl, battery, apps, gps, camera, 2, battery life, gsm, research in motion
toyota prius	hybrid, car, prius, toyota, mpg, photo, price, sales, specs, vehicle, yaris, cars, review, reviews, msrp, specification, milage, model, economy, research
toyota yaris	yaris, hatchback, car, price, hybrid, toyota, models, spec, mpg, liftback, dealer, model, vehicles, review, photo, reviews, city, prius, transmission, vehicle

5.2 Learning

We first compute all probabilities of co-occurences $\Pr(q_1, q_2)$ based on one year of a commercial search engine logs. For each query pair, we use the utility and the implicit task boundary detection models to compute the training targets $\Pr(q_2 \mid q_1, c = 1, u = 1)$, and the features as explained above.

Since training a model on all query pairs would be time consuming, the pairs are subsampled as follows. We first select a subset of queries q_1. Each query q_1 is included in that set with probability $\min(1, c_1 \Pr(q_1))$, where c_1 is a constant. This results in a set of 13,220 queries q_1. Similarly, for each q_1, we select a subset of q_2 with probability $\min(1, c_2 \Pr(q_2 \mid q_1))$. This sampling insures, at both levels, that the head queries are selected as well as some of the tail queries. The median size of q_2 for a given q_1 is 30 and the total number of training instances is 382,740.

The learning algorithm is Gradient Boosted Decision Trees (GBDT) [9]. All the hyperparameters of this algorithm—number of trees, number of nodes, shrinkage factor—are tuned on a separate validation set.

6. SUGGESTION CANDIDATES

We review in this section the three different sources of candidates that will be scored by our model.

6.1 Query logs

The most obvious source of candidates come from the co-occurences in the logs. In order to reduce the number of suggestions, for head queries in particular, we only consider the queries q_2 which co-occured at least 3 times with q_1.

But relying only on query logs still limits the coverage of query suggestions and many queries with a low query log frequency will remain with a few or no suggestions. To further increase the coverage, we use additional sources of information as well, and the rest of this section briefly presents these sources; synthetic suggestions [11], and most frequent specializations.

6.2 Synthetic Suggestions

Here we use a recent work on a synthetic query suggestion generation method that combines a number of unit level operations such as dropping words or several ways of word replacements, to build synthetic query suggestions. Below we give a very brief summary of the method, and for further details please refer to the original paper [11]. Also, note that there is nothing in the relevance model or the feature space that is specifically tuned for this particular method, and one could have used any other query suggestion generation method as well [16, 23]. This is one of the biggest advantages of the proposed framework since it is flexible to work with any type of suggestion candidates since the features do not require historical data from the past session logs.

The first step of the synthetic query generation method we use is a unit importance model, which segments the query into units and assigns importance weights to each unit. Afterwards, less important units of the query are dropped, or replaced with other contextually relevant units. The unit replacements come from sources like: (i) queries that co-occur frequently (same as in PMI, LLR) (ii) phrases from queries that lead to clicks to same URLs ("thanksgiving recipe" and "turkey recipe" lead to clicks on same documents, then "thanksgiving" and "turkey" are substitutable units in the context of "recipe").

Probably the best way to briefly explain how the synthetic query generation works is walking through a few examples. The query "big lots store" does not have many good suggestions as compared to "big lots", due to much lower frequency. Here, after the importance model decides the term "store" can be dropped, it brings the queries that are frequently co-occurring with "big lots" as suggestions for the query "big lots store". The query "cost cutters new jersey" does not have any suggestions, again due to low frequency. The importance model decides "cost cutters" is more important, and drops "new jersey". Queries that frequently co-occur with "cost cutters" are identified and the dropped place name is added back to these to generate synthetic suggestions like "supercuts new jersey", "great clips new jersey". For details on the importance models determines which terms to drop and so on, please refer to the original paper [11].

6.3 Most Frequent Extensions

A significant portion of the query reformulations are extensions, where the user adds more terms to the original query. Even in the cases without a reliable co-occurrence signal, in many cases the most frequent queries that contain the current query can be relevant and useful suggestions. In fact, this idea also has been the backbone of the query completion features in the search industry. The drop-down table that suggests completions of the query as the user is typing is based on the most frequent queries that contain the user entered portion of the query.

We will use up to 20 most frequent queries that contains the original query as a suggestion source, but with one modification; we use a word boundary condition to bring these completions where the query is a full word, so that if the query is “awk” it is not completed to “awkward family pictures” or “awkward tv moments” as it would in query completion, but rather bring suggestions like “awk example” or “awk tutorial”.

Although it does not bring nonsensical queries as in the synthetic suggestions method, this can also bring many irrelevant suggestions and again cannot be used directly. For example for the query “dream”, among many good suggestions like “dream interpretations” or “dream dictionary” this also brings “dream theater”, a progressive rock band that has nothing to do with the meaning of the original query. We use the machine learning model to get rid of such irrelevant suggestions.

7. EVALUATION

We conducted a careful, thorough analysis of the proposed system to verify its effectiveness. In this section, we first review the experimental setup and then offer the empirical analysis through the discussion of the major findings.

7.1 Experimental Setup

Query Set: We collected a random sample of 912 fully anonymized queries issued on a commercial search engine according to their frequency.

Manual Annotation: All manual annotation tasks described were performed by a group of eight professional search engine quality evaluators experienced with assessing the quality of query suggestions and search results.

Annotation Guidelines: Professional annotators provided 4-level ratings (excellent, good, fair, bad) for the ranked suggestions for these 912 unique queries. Annotators were asked to base their judgement after looking at the results page and comparing those for the query and the suggestion; this is important to capture the utility of a suggestion.

Variations of the system: We designed a systematic evaluation where we tested different versions of our system against the baseline. We briefly review each version below:

`B`: As a baseline, the candidates are ranked according to mutual information (2). Only the suggestions with a score larger than 50 are kept, and the threshold value is set empirically by optimizing the trade-off between the quality and coverage.

`10M`: Score of a suggestion is directly computed by the utility estimation $\Pr(q_2 \mid q_1, u = 1, c = 1)$. There is no machine learning model and only the suggestions from the query logs are considered. Co-occurrence of all query pairs are considered within a session of 10 minutes, as described in Section 4.

`10M-ML`: Scores are predicted by the GBDT model. Only the suggestions from the query logs are considered

`10M-ML-SY`: Same as `10M-ML` with the difference that synthetic suggestions (section 6.2) and most frequent specializations (section 6.3) are also scored.

`10MC, 10MC-ML, 10MC-ML-SY`: Same as the above three models except that only consecutive pairs in a session are treated as co-occurrences -to avoid the intent drift in the sessions.

We did not consider a baseline trained on the labeled data alone as an interesting baseline to report on. The reasons can be summarized as follows. With the exception of query substitutions work of Jones et. al. [13], most notable query suggestion research focused on sources other than editorial labels for training machine learning systems. Editorial data is expensive, and generalizing to the tail at web scale requires a lot of labeled data. Search result ranking is probably a simpler problem than query suggestion ranking since, the latter attempts to learn utility left over from the current search. Yet, state of the art methods for learning to rank search results utilize tens of thousands of labeled examples (see Yahoo! Learning to Rank Challenge [4]). Learning to suggest properly would probably require more. Jones et. al. can do with a training set of 1000 query pairs because their task is bid term generation which is much simpler than suggesting queries for various reasons. First, the task is designed to retrieve query substitutions for the current search (so ad coverage can be increased), rather than utility for subsequent searches. Second, non-sensical substitutions are filtered by matching to the advertisers’ bid terms. And last but not least, even the surviving ones are not shown to the user directly. Training on the session data holds other advantages as well. User feedback on additional utility of a suggestion given the current search is directly captured, which is difficult in the editorial data. Our system, on the other hand, does not rely on a large set of labeled data but rather simulates target labels based on the likelihood of the suggestions.

Offline Evaluation: The offline analysis relies on the editorial judgments and there are two major criteria that we deem important to measure. First is the ability of the system to rank the good quality suggestions higher. The second measures the number of queries for which the system is not able to bring any suggestions. These two criteria is analogous to the precision-recall tradeoff in standard document retrieval problems. We adopt DCG and Precision at various cut-off points to measure the quality of the ranked suggestions.

$$
\begin{aligned}
\mathrm{DCG}@k &= \sum_{i=1}^{k} \frac{2^{l_q^i} - 1}{\log(1+i)} \\
P@k &= \frac{\sum_{i=1}^{k} r_q^i}{k}
\end{aligned}
\tag{10}
$$

where $l_q^i \in \{0, 1, 2, 3\}$ is the graded relevance (3 is the best) of the suggestion ranked at position i for a given query q. r_q^i is the binary version where the good and excellent labels are transformed into 1 and the rest is assigned 0. The overall DCG and Precision are calculated by averaging over all the queries in the test data.

We defined the *coverage* as the ratio of the number of queries the system could bring suggestions for to the total number of test queries. Coverage measures the likelihood of a system to find suggestions for a given query. Therefore, it complements the DCG and Precision metrics.

Online Evaluation: For the online analysis, we conducted an A/B test where we tested and compared the proposed system (`10M-ML-SY` implementation) against the baseline on live traffic. Both systems are tested on randomly sampled user populations without imposing any bias towards a particular group of users. The suggested queries appear as related searches at the bottom of the result page. Users interactions

Table 5: Coverage, DCG, and Precision improvement compared to the mutual information baseline.

	depth = 1	depth=3	depth=5	depth=7	depth=9	depth=12
Coverage						
`10M`	-9%	2%	2%	4%	34%	294%
`10MC`	-9%	0%	0%	2%	33%	282%
`10M-ML`	-13%	-3%	-1%	2%	32%	289%
`10MC-ML`	-17%	-7%	-6%	-4%	25%	269%
`10M-ML-SY`	17%	28%	35%	40%	85%	381%
`10MC-ML-SY`	17%	28%	34%	40%	84%	365%
DCG on common coverage						
`10M`	2%	5%	5%	6%	9%	4%
`10MC`	2%	9%	8%	8%	9%	7%
`10M-ML`	5%	8%	8%	8%	9%	10%
`10MC-ML`	5%	7%	8%	9%	10%	9%
`10M-ML-SY`	7%	9%	8%	9%	10%	10%
`10MC-ML-SY`	8%	8%	7%	10%	12%	11%
Precision on common coverage						
`10M`	4%	10%	9%	10%	14%	11%
`10MC`	8%	13%	11%	12%	15%	11%
`10M-ML`	7%	10%	10%	11%	12%	12%
`10MC-ML`	6%	8%	10%	13%	13%	12%
`10M-ML-SY`	8%	11%	10%	11%	13%	14%
`10MC-ML-SY`	8%	11%	10%	13%	16%	16%

Table 6: Coverage and CTR improvements of the `10M-ML-SY` system in the online test.

	depth = 1	depth=3	depth=5	depth=7	depth=9	depth=12
Coverage	10%	31%	40%	40%	35%	91%
CTR	42%	41%	38%	37%	50%	49%

with the suggested queries are logged in the query logs for a period of one week. To ensure production quality and sufficient diversity of the suggestions, we further remove the low utility queries that are near-duplicates of already suggested queries. The duplicate elimination method is identical in both baseline and test buckets; the details of the model is explained in detail in [18]. We measured the CTR on the suggestions that our system ranked and compared it against the CTR on those ranked by a MI baseline. Here CTR at position k is defined as number of clicks within the first k suggestions divided by number of result pages with at least k suggestions.

A/B tests are expensive in the sense that they require engineering resources to build the necessary online platform to do a full comparison. For this reason, instead of implementing online tests for each variation of the model, for the online test we picked `10M-ML-SY`, one of the best performing models in the offline comparisons, with respect to coverage and the quality metrics based on human judgments.

7.2 Results

In Table 5, we show the relative performance of different versions of the proposed system against the baseline with respect to DCG, Precision and Coverage metrics at various cut-off points (depths). The DCG and Precision values are calculated on the queries that both systems bring suggestions at the specified depth; hence we refer to it as common coverage in Table 5. Generally, the performance gap increases with depth where the largest gaps occur at depths 9 and 12. At depths 9 and 12, the best performer in terms of DCG against the baseline is `10MC-ML-SY` followed by `10M-ML-SY` and `10M-ML`. The same is also true in terms of Precision. This suggests that i) the machine learned models outperform the systems that directly uses estimated target scores $\Pr(q_2 \mid q_1, u = 1, c = 1)$ without any learning, ii) the system is able to blend the additional candidates very well that the overall quality of the suggestions are as good as the organic counterpart. Furthermore, the performance of the proposed approach (in both DCG and Precision) either outperforms or is comparable to the strong MI (Mutual Information) baseline at all depths.

The coverage remains relatively flat across different methods at small depths. This is not surprising since all methods are effective in bringing at least a few suggestions for a given query. However, the coverage difference becomes more evident at larger depths where the real benefit of the implicit task boundary model and the machine learning step can be assessed. At depths 9 and 12, the best coverages are those of `10M-ML-SY` and `10MC-ML-SY`. We are quite encouraged by this result since it demonstrates that our approach can blend the additional candidates into organic ones without hurting the performance while maintaining a high coverage. The coverage of `10MC-ML` is quite low compared to these two, since the higher quality is suffered by a loss in coverage for `10MC-ML`. The coverage of the baseline, on the other hand, is significantly lower than the rest at larger depths even though it has comparable coverage to others at smaller depths.

In Figure 1 we demonstrate how the editorial judgments are distributed for the baseline, and the full machine learning model `10M-ML-SY` that is used in the online test. To be able to show how much of the gain comes from the implicit task boundary model, and how much of it comes from the machine learning step, we present the grade distributions of the implicit task boundary model `10M` as well. We compared the grade distribution at different cut-off points from top 1 to top 12. Also, the grades presented in the histograms are computed over the each individual coverage of the methods, hence they are different than those in 5, which compares the relevance over the common coverage. A few things to note in Figure 1 are: (i) `10M` and `10MC-ML-SY` do perform better not only with respect to the common coverage (as presented in Table 5) but also over their entire coverage, which is significantly bigger than the common coverage with the baseline. (ii) At every rank cut-off, the ratio of good and excellent suggestions are higher and the ratio of fair and bad suggestions are lower for both `10M` and `10MC-ML-SY`, hence the grade distributions are more skewed towards better grades. (iii) Although the implicit task boundary model `10M` is better than the baseline in terms of total number of good or excellent suggestions, the machine learning step has a more important contribution in bringing a lot more good quality suggestions without increasing the ratio of bad suggestions.

The results of the online analysis are similarly encouraging. We compared our system against the MI baseline in terms of both CTR and coverage. We note that the definition of coverage and CTR in the online evaluation is slightly different that those used in the offline analysis. In the online version, the coverage is weighted by query frequency. On the other hand, the click ratios are calculated separately for each system on the respective queries where there is a suggestion. The results are presented in table 6. They show that our system not only increases the chance of finding suggestions for a given query, it also ensures a significantly higher quality of these suggestions. Additionally, we report a 0.9% decrease in the next-page clicks using our system. Users' needs for a suggestion are higher at the bottom of a page since they view the top 10 results before they see the related searches at the bottom. The fact that the CTR increases and the next-page clicks decreases indicate to some extent that the suggestions were useful.

To understand what the model does, it is instructing to look at which features were the most relevant in the GBDT model. Overall result set features are much more important than lexical features, with `ABOUTNESS` and some LTR based scores being the most relevant ones. This is not surprising since the result set features are more sophisticated features. Among the lexical features `LEV` and more generally features that depend on both q_1 and q_2 were more heavily used by the GBDT model.

Finally Table 7 lists the results of the two models for a few sample queries.

8. CONCLUSIONS

In this paper we present an end-to-end query suggestion method that implements novel ideas such as incorporating usefulness of reformulations, an implicit session boundary model, and a machine learning model to further improve the suggestion relevance and be able to add more sources of suggestions beyond the co-occurrences in query logs. The idea of incorporating usefulness into the query co-occurrence models is not specific to our particular selection of target, and it can be used directly with PMI, LLR, and other similar measures as well. Even without the machine learning step, the reformulation relevance model shows significant improvements over MI, significant gains in coverage and modest gains in relevance. The feature generation and machine learning step brings some further relevance improvement, and allows us to add other candidate sources, which significantly increases the coverage. This is a particular advantage of our approach against the related work in the literature. We trained the machine learning model with targets generated from session logs via the utility and implicit task boundary models, which removes the dependency on large labeled data. The offline as well as the online evaluation demonstrate the effectiveness of the proposed framework against MI. We observed significant improvements in coverage and quality metrics. The click through rates on the online tests are very promising and we plan to extend this work with personalization and further diversification of suggestions.

Table 7: sample queries and their suggestions for `10M` and the baseline

query	odd couple
`B`	tony randall
`10-M`	jack klugman, tony randall, odd couple movie, tony curtis, youtube, john fiedler, jack lemmon, imdb, grumpy old men, odd couple theme, taxi
query	al di meola
`B`	al di meola discography, al di meola tabs, john mclaughlin, paco de lucia, twitter
`10-M`	john mclaughlin, youtube, stanley clarke, paco de lucia, lee ritenour, return to forever, jeff beck, amazon, al di meola youtube
query	skateboard wheels
`B`	cheap skateboard wheels, spitfire skateboard wheels
`10-M`	longboard wheels, ccs, skateboard, skateboards, amazon, cheap skateboard wheels, skateboard trucks, skateboard decks, soft skateboard wheels, big 5, purple skateboard wheels, sports authority
query	sigir
`B`	-
`10-M`	sigar, sigir 2012, special inspector general for iraq, sigir current news
query	Balvenie
`B`	balvenie scotch
`10-M`	glenlivet, glenfiddich, balvenie doublewood, glenmorangie, talisker, macallan, balvenie scotch, lagavulin, scotch
query	wool sweater
`B`	-
`10-M`	ll bean, women wool sweater, women's wool sweater, ping pong table

9. REFERENCES

[1] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 407–416. Acm Press, 2000.

[2] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 609–618, New York, NY, USA, 2008. ACM.

[3] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *Proceedings of the 2009 workshop on Web Search Click Data*, WSCD '09, pages 56–63, New York, NY, USA, 2009. ACM.

[4] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. *Journal of Machine Learning Research - Proceedings Track*, 14:1–24, 2011.

[5] L. B. Chilton and J. Teevan. Addressing people's information needs directly in a web search result page. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 27–36, 2011.

[6] H. Deng, I. King, and M. R. Lyu. Entropy-biased models for query representation on the click graph. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 339–346, New York, NY, USA, 2009. ACM.

[7] T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19:61–74, March 1993.

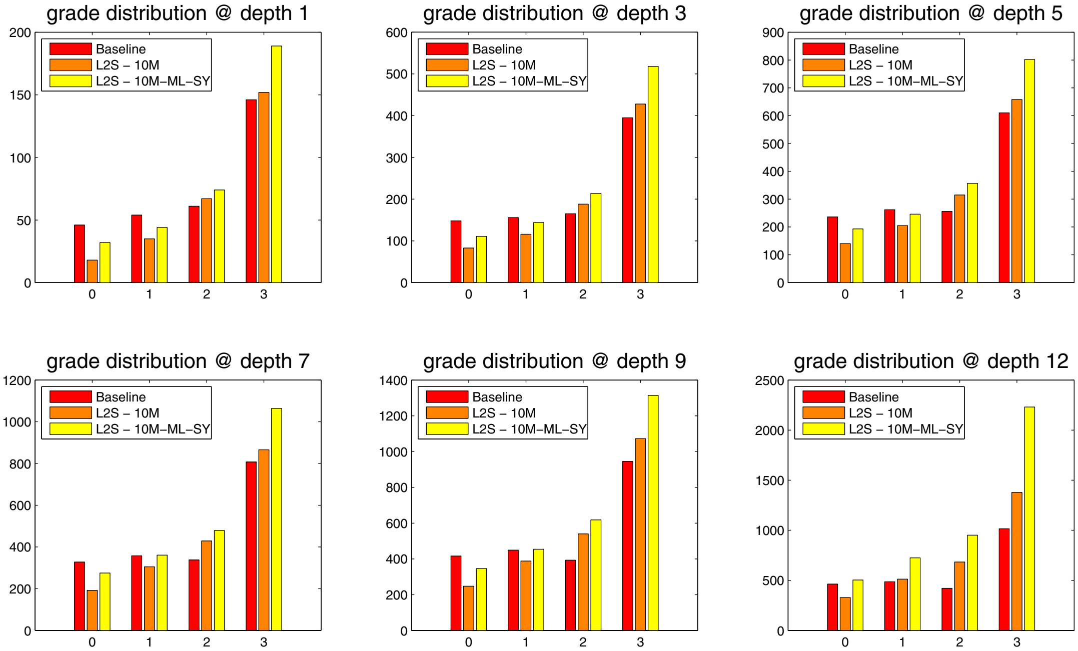

Figure 1: Grade distributions (3=Excellent, 2=Good, 1=Fair, 0=Bad) at different ranks for `10M` and `10M-ML-SY` versus the baseline.

[8] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: social searching? In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '97, pages 306–313, New York, NY, USA, 1997. ACM.

[9] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29, 2000.

[10] C. Huang, L. Chien, and Y. Oyang. Relevant term suggestion in interactive web search based on contextual information in query session logs. *Journal of the American Society for Information Science and Technology*, 54:638–649, 2003.

[11] A. Jain, U. Ozertem, and E. Velipasaoglu. Synthesizing high utility suggestions for rare web search queries. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, SIGIR '11, pages 805–814, 2011.

[12] E. C. Jensen, S. M. Beitzel, A. Chowdhury, and O. Frieder. Query phrase suggestion from topically tagged session logs. In H. L. Larsen, G. Pasi, D. O. Arroyo, T. Andreasen, and H. Christiansen, editors, *Flexible Query Answering Systems, 7th International Conference, FQAS 2006, Milan, Italy, June 7-10, 2006, Proceedings*, volume 4027 of *Lecture Notes in Computer Science*, pages 185–196. Springer, 2006.

[13] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 387–396, New York, NY, USA, 2006. ACM.

[14] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, (10), 1966.

[15] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 469–478, New York, NY, USA, 2008. ACM.

[16] G. A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41, 1995.

[17] R. C. Moore. On Log-Likelihood-Ratios and the Significance of Rare Events. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*, 2004.

[18] U. Ozertem, E. Velipasaoglu, and L. Lai. Suggestion set utility maximization using session logs. In *Proceedings of the 20th international ACM Conference on Information and Knowledge Management*, CIKM '11, 2011.

[19] D. Paranjpe. Learning document aboutness from implicit user feedback and document structure. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 365–374. ACM, 2009.

[20] V. V. Raghavan and H. Sever. On the reuse of past optimal queries. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 344–350. ACM Press, 1995.

[21] C. H. Ricardo Baeza-Yates and M. Mendoza. Query recommendation using query logs in search engines. In *Trends in Database Technology - EDBT 2004 Workshops*, 2005.

[22] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 841–850, New York, NY, USA, 2010. ACM.

[23] I. Szpektor, A. Gionis, and Y. Maarek. Improving recommendation for long-tail queries via templates. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 47–56, New York, NY, USA, 2011. ACM.

[24] A. Thanopoulos, N. Fakotakis, and G. Kokkinakis. Comparative evaluation of collocation extraction metrics. In *Proceedings of the 3rd Language Resources Evaluation Conference*, pages 620–625, 2002.