Searchable Web Sites Recommendation

Yang Song
Microsoft Research
One Microsoft Way
Redmond, WA
yangsong@microsoft.com

Nam Nguyen
Microsoft Corp.
One Microsoft Way
Redmond, WA
namn@microsoft.com

Li-wei He
Microsoft Research
One Microsoft Way
Redmond, WA
lhe@microsoft.com

Scott Imig
Microsoft Corp.
One Microsoft Way
Redmond, WA
scottim@microsoft.com

Robert Rounthwaite
Microsoft Research
One Microsoft Way
Redmond, WA
robertro@microsoft.com

ABSTRACT

In this paper, we propose a new framework for searchable web sites recommendation. Given a query, our system will recommend a list of searchable web sites ranked by relevance, which can be used to complement the web page results and ads from a search engine. We model the conditional probability of a searchable web site being relevant to a given query in term of three main components: the language model of the query, the language model of the content within the web site, and the reputation of the web site searching capability (static rank). The language models for queries and searchable sites are built using information mined from client-side browsing logs. The static rank for each searchable site leverages features extracted from these client-side logs such as number of queries that are submitted to this site, and features extracted from general search engines such as the number of web pages that indexed for this site, number of clicks per query, and the dwell-time that a user spends on the search result page and on the clicked result web pages. We also learn a weight for each kind of feature to optimize the ranking performance. In our experiment, we discover 10.5 thousand searchable sites and use 5 million unique queries, extracted from one week of log data to build and demonstrate the effectiveness of our searchable web site recommendation system.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Search process

General Terms

Algorithms, Experimentation

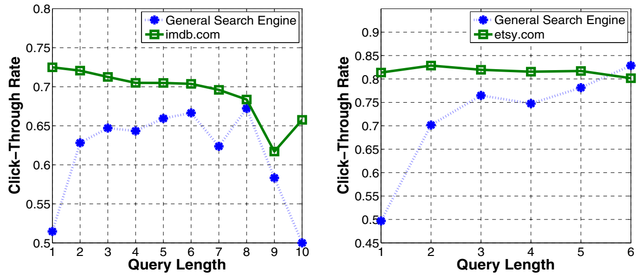


Figure 1: Comparison of Click-Through Rate (CTR) of a general search engine and site search. Searching in web sites directly results in significantly higher CTRs.

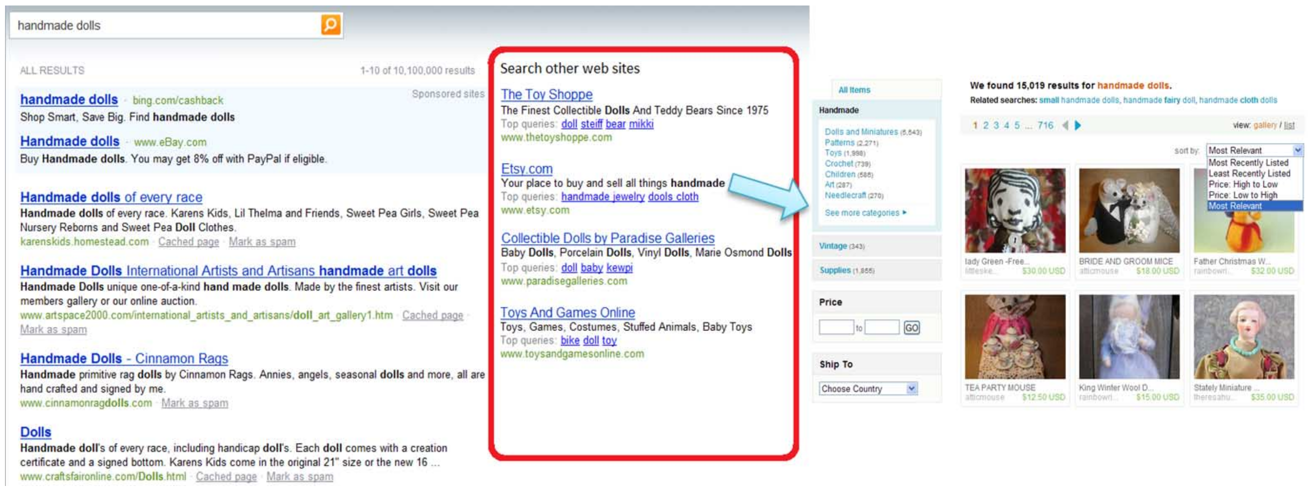
Keywords

Vertical Search Engines

1. INTRODUCTION

In addition to popular general search engines such as Google, Bing and Yahoo, many commercial and non-commercial web sites offer searching capability for specialized content: e.g. imdb.com for movie-related topics, webmd.com for health-related questions, walmart.com for things to buy, etc. Even though general search engines are able to index a tremendous number of web pages and have increasingly sophisticated ranking algorithms, performing searches on web sites directly often have distinct advantages:

- Search engines may have no or limited access to proprietary content or database;
- More domain-specific information can be used for ranking the results. For example, given a query "pinot noir", a wine web site might return the wines ranked by taste score, price or both;
- The search results are often presented with a more appropriate format or user interface;
- Since each web site usually focuses on a specific domain, the user has to do less filtering of totally unrelated results.


handmade dolls

ALL RESULTS 1-10 of 10,100,000 results

Sponsored sites

handmade dolls · bing.com/cashback
Shop Smart, Save Big. Find **handmade dolls**

Handmade dolls · www.eBay.com
Buy **Handmade dolls**. You may get 8% off with PayPal if eligible.

Handmade dolls of every race
Handmade dolls of every race. Karens Kids, Lil Thelma and Friends, Sweet Pea Girls, Sweet Pea Nursery Reborns and Sweet Pea **Doll** Clothes.
karenskids.homestead.com · Cached page · Mark as spam

Handmade Dolls International Artists and Artisans handmade art dolls
Handmade Dolls unique one-of-a-kind **hand made dolls**. Made by the finest artists. Visit our members gallery or our online auction.
www.artspace2000.com/international_artists_and_artisans/doll_art_gallery1.htm · Cached page · Mark as spam

Handmade Dolls - Cinnamon Rags
Handmade primitive rag **dolls** by Cinnamon Rags. Annies, angels, seasonal **dolls** and more, all are hand crafted and signed by me.
www.cinnamonragdolls.com · Mark as spam

Dolls
Handmade doll's of every race, including handicap **doll**'s. Each **doll** comes with a creation certificate and a signed bottom. Karens Kids come in the original 21" size or the new 16 ...
www.craftsfaironline.com/Dolls.html · Cached page · Mark as spam

Search other web sites

The Toy Shoppe
The Finest Collectible **Dolls** And Teddy Bears Since 1975
Top queries: doll steiff bear mikki
www.thetoyshoppe.com

Etsy.com
Your place to buy and sell all things **handmade**
Top queries: handmade jewelry dools cloth
www.etsy.com

Collectible Dolls by Paradise Galleries
Baby **Dolls**, Porcelain **Dolls**, Vinyl **Dolls**, Marie Osmond **Dolls**
Top queries: doll baby kewpi
www.paradisegalleries.com

Toys And Games Online
Toys, Games, Costumes, Stuffed Animals, Baby Toys
Top queries: bike doll toy
www.toysandgamesonline.com

We found 15,019 results for handmade dolls.
Related searches: small handmade dolls, handmade fairy doll, handmade cloth dolls

Figure 2: Given a query *handmade dolls*, our proposed interface recommends list of links to most relevant searchable sites. Clicking on the link directs the user to the search result page of a recommended site of the query *handmade dolls*.

Figure 1 demonstrates the click-through rates (CTR) of imdb.com and etsy.com as compared to a general search engine. For the same set of queries, users are much more likely to make clicks when they issue queries on domain-specific web sites than on a general search engine. The problem is: users often do not know which domain-specific sites to go to. Therefore, our goal in this paper is to discover, rank and recommend relevant searchable web sites to users. Figure 2 demonstrates an example of proposed interface, where the searchable web sites are listed on the right-hand side of regular search results. Clicking on the link directs the user to the search result page of a recommended site of the query *handmade dolls* with much richer content information and ranking capability.

In recent years, there has been much attention to the problem of user intent classification in information retrieval [15, 12, 11, 7, 13, 2, 5, 10, 19]. Previous works focus on categorizing a given query into a general intent such as news, image, autos and so on, which can be identified as macro vertical/intent selection. For example, in [2], the authors empirically selected 18 topics and trained 19 binary classifiers for query categorization. The reason for previous approaches to pick only high-level categories is because most of them belong to supervised learning methods, which require careful human-labeling of training data. e.g., in [2], 25,000 queries are labeled by human judgers. These approaches are thus very expensive and unable to make prompt response to topic evolution over time.

Compared to existing approaches, the novelty of our method is twofold. First, instead of using queries from server-side search engine logs, we are among the first to *leverage user queries from client-side browsing logs* of a widely-distributed web browser. The client-side logs offer a rich set of information which can be leveraged for site activity detection. e.g., in-site queries, dwell-time on a page, user click sequence, unique number of users on a site and so on.

Secondly, we use a *completely unsupervised* approach for searchable site recommendation. Our method does not involve hand-labeling of any data or features. The model is capable of automatically extracting features from logs and learning optimal weights for each feature. With the ability to adapt itself, our method can continuously refine the model over time and therefore scale well with terabytes of data per day.

Specifically, this paper makes the following contributions:

- Using a simple pattern matching approach, we are able to discover more than 10,000 searchable web sites from one-week's client-browsing logs. Overall, users submitted more than 5 million queries to these sites with over 20 million click activities.
- Given a set of extracted features, a Bayesian language model is proposed for site selection, which models the conditional probability of a site being relevant to a given query and outputs a ranked list of relevant sites.
- An effective query expansion technique is proposed to model the conditional probability of words being relevant to a given query, which successfully alleviates the sparseness issue during model learning and exhibits significant improvement in recommendation quality.
- Empirical results on one-week's browsing log demonstrate the effectiveness of our proposed method in a large-scale setting where there are thousands of searchable web sites and millions of queries.

The rest of the paper is organized as follows. Section 2 discusses related work for the problem of intent classification. Section 3 describes the data collection procedure and the web site discovery method. Section 4 introduces our learning framework for the searchable site recommendation problem and our proposed solution. Section 5 describes our data preparation as well as the experiment setup and result. Finally, we discuss some advantages of our approach and give the concluding remarks in Section 6 and Section 7, respectively.

2. RELATED WORK

There is a large body of recent work on the problem of user intent classification. Here we simply highlight some of

Table 1: The key differences between our approach and previous arts.

	Previous Works	Our Work
Cost of learning	Mainly supervised approach, - hand-labeled queries for binary classification [2] - used Wikipedia categories for query intent [13]	Completely unsupervised model - automatic feature extraction - scale up to Terabyte of data
Granularity of the model	Limited number of verticals - focused on topic level macro verticals - maximum reported number is 18 [2]	Large number of potential searchable sites - drilled down to individual sites - discovered more than 10,000 sites
Data source	Mainly search engine click logs - Only click-level features	Client-side browsing logs - complete browsing trails with many features

the more relevant works. In [13], the authors addressed the problem of query intent classification by leveraging the concepts defined by Wikipedia as a representation space. Each intent domain is represented as set of Wikipedia articles and categories. For each intent, the authors first identified a few Wikipedia categories that belong to the intent. Secondly, the initial set of categories is extended by applying random walk on the predefined Wikipedia link graph. The intent of any input query is determined by searching for the closest match in the Wikipedia representation space. In comparison to our approach, the method proposed by [13] requires hand-labeled data for each intent, hence it is not effective to apply to a large-scale system with thousands of intents.

In [2], the authors addressed the problem of vertical selection, predicting a relevant vertical (if any) for a given query. The proposed method first constructs a feature representation for each query based on three sources of evidence: (1) the query string, from which features are derived independent of external resources, (2) logs of queries previously issued directly to the vertical, and (3) corpora representative of vertical content. Secondly, the method learned a binary classifier for each of the 18 different verticals and one additional classifier to indicate no relevant verticals by using about 25 thousand hand-labeled training queries. In comparison to our approach, the method proposed by [2] also needs a large amount of hand-labeled training data in order to be effective. Moreover, the authors only experimented with 18 verticals versus thousands of searchable sites in our experiment. In addition, in [11], the authors extended the vertical selection problem to incorporate user feedback.

In [15], the authors proposed a method to improve performance of query intent classifiers by taking advantage of click graphs. In this work, the method infers class memberships of unlabeled queries from those of labeled ones according to their proximities in a click graph. The authors also regularized the inferred labeled with click graphs by content-based classification to avoid propagating erroneous labels. Similar to our approach, [15] is able to exploit the query-vertical relationship in the user log data via the click graphs to expand the amount of training data. However, the proposed method still need some labeled training examples for each intent and the paper only experiments with two different intents: job intent and product intent.

In [12], the authors proposed an unsupervised learning method to the problem of query categorization. The model is stored as a graph of concepts where graph edges represent the cross-reference between the concepts. Similar to our approach, the authors used a general search engine to build the concepts and their relations. To determine the category of a new query, the proposed method first discovers the relevant nodes in the concept graph which then can be used to map the query to the related category.

In [7], the authors surveyed and evaluated some of the query intent detection methods on three main underlying intents, namely navigational, informational, and transactional. Even though the work does not include some of the more recent methods on vertical selection, it does point out an important issue about evaluation method. In our experiment, with the lack of manually prepared gold-standard labeled query-intent data, we has proposed an empirical evaluation method based on the user log data.

[8] is an application of query intent detection. In [8], the authors proposed a method for bidterm suggestion based on the query intent. The proposed method models an advertiser's intent and finds new bidterms consistent with that intent.

In [6], the author demonstrated the importance of the specialized vertical search engines in term of retrieval task. [6] identified domain-specific search strategies in the healthcare and shopping categories and showed that expert knowledge is not automatically obtained from using general purpose search engines.

As an alternative approach, [17] proposed a method to crawl and index the Deep-Web content. Searchable web sites are a significant portion of the Deep-Web since they usually only provide search HTML form to query their content. In [17], the authors presented an algorithm for selecting input values for text search inputs that accepts keywords and an algorithm for identifying inputs which accept only values of a specific type. A similar approach was taken by [4] where a two step filtering algorithm was used to discover relevant online databases (forms). On the other hand, authors in [9] built a keyword-based decision tree to discovery searchable webpages. The rules and keywords in decision tree are fixed and cannot be adjusted to individual pages easily. In comparison to our approach, [17], [4] and [9] require tremendous amount of computing power as well as prior domain knowledge and does not take advantage of users' knowledge. Thus, large searchable web sites could take very long time to crawl and index.

In Table 1, we summarize some characteristics of previous work that we are addressing in this paper.

3. DATA COLLECTION

First, we describe the format of the data that is used to learn our model. The log data used in this paper is the client-side browsing log collected from millions of opt-in users of a world-wide distributed browser. In the log data, we record all users' activities within the browser. For each user, the activities are divided into sessions if there is no activity within

Table 3: **The most commonly used query parameters for searchable web sites.**

Parameter	Examples
query=	webmd.com/default.aspx?query=Tdap
q=	imdb.com/find?q=transformers
keywords =	overstock.com/search?keywords=boots
keyword =	footlocker.com/?keyword=running+shoes
SearchTerm=	bebo.com/SearchTerm=mj
st=	washingtonpost.com/Search?st=h1n1
w=	clothing.bodenusa.com/search?w=bags
terms =	herroom.com/Search2?terms=mens+bra
searchtext=	www.igigi.com/?searchtext=weekend+clothes
qt=	search.ama-assn.org/qt=flu+shot

30 minutes. Each activity in a session records the following information[1]: unique user id, clicked url, referred url, IP address, time stamp and the sequential order within a session (vis_seq). All activities within a session are organized according to their time stamps. Specifically, the referred url is the one that current url is clicked from. Therefore, by following the referrer urls, we can further separate the user activities into different trails which can be recognized as *user tasks*. For example, a user may open a tab of the browser to read news while checking emails in another. For the purpose of this paper, we divide the tasks into *search trails* and *non-search trails*. An example of these two trails are shown in Table 2, where the top trail shows user search on a general search engine and the bottom on a more specialized web site. For discovering potential searchable sites which are unknown (or ranked lower than popular sites) by general search engines, it makes sense to leverage only the non-search trails from the logs.

A simple pattern matching rule is used to extract queries from clicked URLs in the logs. For most web sites, the search keywords are passed as parameters in URLs such as “query=”. Different parameters are separated using “&” character. In this paper, we only focus on the most discriminative parameter for each specialized engine. So for a URL containing two parameters “query=audo+a6&zip=98034”, only the “query” part is extracted. In this paper, a set of 126 manually extracted patterns are used. The determination of the most relevant parameter is beyond the scope of this paper. In Table 3, we summarizes the most commonly used search keywords.

Overall, we collected one-week’s browsing log from July 1 to July 7 2009 for training purpose. The data is over 20 Terabytes, with a total of 5 billion web page clicks. Figure 3 plots the transition probability between search engine result pages (SERP) and common web pages (URLs). Comparing to over 86% of times which users browse from one page to another, the search activity is only a tiny part of the client-side log. Using our simple pattern matching criteria, we were able to discover 10,546 potentially searchable sites. The total unique queries submitted to sites directly are over 5 million. Noticeably, we intentionally removed a few dozen well-known searchable sites from our list (including youtube.com, ebay.com, craigslist.org and so on). We choose not to recommend them because they are so popular

[1]There are many other unrelated fields that are also recorded. As a preliminary study, we only utilize these meta features.

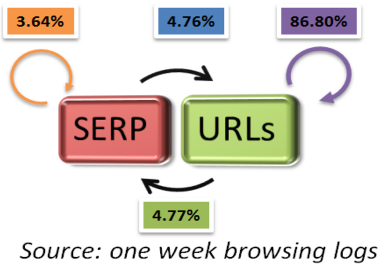


Figure 3: The transition probability between two states: search engine result pages (SERP) and common URLs (URLs). The self transition probability of URLs consists the majority of user web browsing.

Table 4: Top 10 frequently searched web sites according to query frequency.

Web Site	Total Searches	Total Clicks
www.indeed.com	824543	2734716
www.walmart.com	505516	1715079
www.flickr.com	416617	1615179
www.imdb.com	393710	1367888
www.hulu.com	351040	1342549
www.myxer.com	341967	1239722
www.amazon.com	304055	868667
www.etsy.com	294507	597705
www.linkedin.com	285228	1057392
mp3.zing.vn	267627	860819

that most general search engines have already performed a good job of recommending these sites when relevant queries are issued, so that we concentrate on the less popular ones in this paper, because generally people have more trouble finding these sites. Saying this, the performance of these top web sites will be discussed in the Discussion section to favor our proposed technique.

Table 3 lists the top searched sites according to query frequency. The topics of discovered web sites cover the most macro verticals addressed in previous works [2].

4. RECOMMENDATION FRAMEWORK

In contrast to previous work, we model the searchable web site recommendation as a ranking problem instead of classification. Given a query, our system provides a ranked list of best-matched searchable web sites. In order to model the conditional probability of engines given queries, we construct a set of features that can be automatically extracted from the logs.

4.1 Feature Generation

The first step of feature generation is to extract searchable web sites as well as the user queries. Specifically, we extract

- A set of web sites users issued queries to, denoted as V.
- A set of queries per web site, denoted as Q_v for a site $v \in V$.

A set of seven descriptive features for each searchable site are constructed. These features are used to estimate the

Table 2: An example of user search trails on general search engines (Top) and a searchable web site. (Bottom).

userid	clicked url	referred url	vis_seq	local time
UID	www.bing.com	-	1	10:10:17.281-04:00
UID	www.bing.com/search?q=dolls	www.bing.com	2	10:10:54.531-04:00
UID	karenskids.homestead.com	www.bing.com/search?q=dolls	3	10:11:20.484-04:00
UID	www.craftsfaironline.com/Dolls.html	www.bing.com/search?q=dolls	4	10:11:27.953-04:00
UID	www.bing.com/search?q=cheap+dolls	-	5	10:11:36.390-04:00
UID	american-doll-clothes.com	www.bing.com/search?q=cheap+dolls	6	10:11:58.187-04:00
...				
UID	www.etsy.com/	-	15	10:40:43.515-04:00
UID	www.etsy.com/search_query=dolls	www.etsy.com/	16	10:40:48.578-04:00
UID	www.etsy.com/listing_id=32079621	www.etsy.com/search_query=dolls	17	10:41:20.125-04:00
UID	www.etsy.com/listing_id=33551504	www.etsy.com/search_query=dolls	18	10:41:45.343-04:00
UID	www.etsy.com/cartcheckout?id=33551504	www.etsy.com/listing_id=33551504	19	10:41:50.109-04:00

reputation of each web site's *searching capability*, which is different from simply the site's PageRank. Instead, they measure how much people use the site's search feature and how satisfied they are after each use.

1. the total number of queries submitted to the site (the cardinality of Q_v), denoted as queries(v), which indicates the popularity of the site's search feature,

2. the number of unique queries submitted to the site, denoted as unique_queries(v), which is a measure of the popularity and topic coverage of the site's search feature,

3. the average number of clicks per query on site v, denoted as clicks(v), which helps identify whether the user's information need is satisfied,

4. the average dwell-time on the search result page per query on v, denoted as dt1(v), which helps to access the quality of the result links,

5. the average dwell-time on the referred links per query on v, denoted as dt2(v), which helps to measure the quality of the retrieved information,

6. the number of web pages belonging to the site that are indexed by a general search engine, denoted Index(v), which is an approximation of the quantity of web pages within v, and

7. the entropy of the topic distribution based on the Open Directory Project's (ODP) hierarchy of the web [1], denoted entropy(v), which measures the diversity of the site's content. Specifically, to compute the ODP-topic distribution entropy for each specialized vertical search engine, we utilize the one-versus-all binary classifiers from 219 most popular topics from the ODP hierarchy. The conditional probability of a ODP topic t given a vertical search engine v is defined as

$$P(t|v) = \frac{1}{|Q_v|} \sum_{q \in Q_v} f_t(q^*), \tag{1}$$

where $f_t(q^*)$ is the calibrated classification output of the binary classifier for the ODP-topic t when applying to the expanded query q^*. Hence, the topic entropy of a web site is computed as

$$\text{entropy}(v) = -\sum_t P(t|v) \log P(t|v).$$

Web sites with low entropies are sought to have better specialty, while more general web sites often have higher entropy scores. An example has been shown in Figure 4 for ESPN and Google, where Google has a much higher entropy score than ESPN which mainly focuses on the sport category.

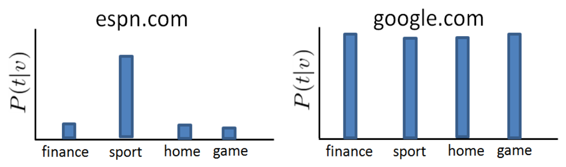


Figure 4: Topic distribution $P(t|v)$ of two sites: ESPN and Google, where more general search engines have flatter topic distribution thus higher entropy scores.

4.2 Components Breakdown

The recommendation problem can be formulated as the conditional probability of a searchable web site v given a query q, $P(v|q)$, as follows:

$$\begin{aligned} P(v|q) &= \sum_{w \in \text{Vocab}} P(w, v|q) \\ &= \sum_{w \in \text{Vocab}} P(w|q) P(v|q, w) \\ &= \sum_{w \in \text{Vocab}} P(w|q) P(v|w) \\ &= \sum_{w \in \text{Vocab}} P(w|q) P(w|v) \frac{1}{P(w)} P(v) \end{aligned} \tag{2}$$

In this equation, we first introduce a new variable w representing the word or term in our vocabulary, which is denoted as Vocab. Hence, the first equation is the summation over all possible values of the variable w of the joint conditional probability, $P(w, v|q)$. In addition, we also make an assumption that the term w and the query string q represent redundant information so that $P(v|q, w) = P(v|w)$. Ideally, queries should be modeled with n-gram language models with $n > 1$ while w represents a simple unigram model. However, since many literature works have shown that higher order of language models can barely help improving the retrieval performance, we are ignoring the order of the terms within the query string in our work. In our simplified representation, we believe that the single term w can effectively explain the information need of a query q in most scenarios. Finally, we apply the Bayes' rule for the conditional probability to get $P(v|w) = \frac{P(w|v)P(v)}{P(w)}$.

As can be observed from eq.(2), the conditional probability of a searchable site v given a query q can be broken down into four components: (1) the conditional probability of a term w given the query q, $P(w|q)$; (2) the conditional probability of the term w given the searchable site v, $P(w|v)$; (3) the inverse of the probability of the term w, $\dfrac{1}{P(w)}$; and (4) the prior belief or the static rank of the searchable site v, $P(v)$. In this paper, we leverage language modeling approaches [22] to each of these four components.

4.2.1 Model P(w|q) via Query Expansion

Given that typical queries submitted to searchable web sites contain very few keywords [14], we expand each query q with the snippets of the top-returned results by submitting the query to a general search engine. Since the snippets on search engine result pages are selected from the most descriptive words of the web page contents, it offers great free resources for better understanding user intents[2], especially when the queries are ambiguous. Specifically, the titles and the descriptive words are both leveraged for query expansion (see Figure 5). In this paper, we empirically use top-50 results from a general search engine for query expansion.

We denote the expanded query as q^*. Hence, our conditional probability of a term w given a query q, $P(w|q)$, is estimated based on the expanded query q^*, $P(w|q^*)$. The language model of $P(w|q^*)$ is modeled as a multinomial distribution, for which the conjugate prior for Bayesian analysis is the Dirichlet distribution [16],

$$P(w|q) \simeq P(w|q^*) = \frac{\text{tf}(w;q^*) + \mu P(w|C)}{\sum_{w' \in \text{Vocab}} \text{tf}(w';q^*) + \mu} \qquad (3)$$

where μ is the Dirichlet smoothing parameter; $\text{tf}(w;q^*)$ is the term frequency of a term w in the expanded query q^*; and $P(w|C)$ is the language model for the entire expanded training query collection,

$$P(w|C) = \frac{\text{tf}(w;C)}{\sum_{w' \in \text{Vocab}} \text{tf}(w';C)}. \qquad (4)$$

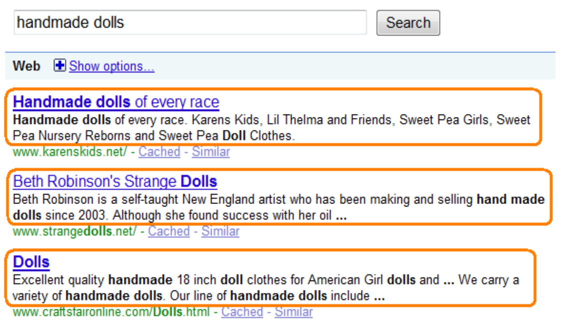
4.2.2 Model P(w|v) and P(w)

In order to model the conditional probability of the term w given the searchable site v, we utilize the set of queries Q_v containing all submitted queries to the site v. The language model for a searchable site v, $P(w|v)$, is computed as the average of all language models of the queries,

$$P(w|v) = \frac{1}{Z} \sum_{q \in Q_v} P(w|q)\mathcal{I}(\#\text{ of click}(v,q) > 0) \qquad (5)$$

where $Z = \sum_{q \in Q_v} \mathcal{I}(\#\text{ of click}(v,q) > 0)$ is a normalizing constant. In this formula, we only include queries whose search

[2]Similarly, in [13] the authors also used Wikipedia as an external source for representation of the query intent.



Handmade dolls of every race
Handmade dolls of every race. Karens Kids, Lil Thelma and Friends, Sweet Pea Girls, Sweet Pea Nursery Reborns and Sweet Pea **Doll** Clothes.
www.karenskids.net/ - Cached - Similar

Beth Robinson's Strange **Dolls**
Beth Robinson is a self-taught New England artist who has been making and selling **hand made dolls** since 2003. Although she found success with her oil **...**
www.strange**dolls**.net/ - Cached - Similar

Dolls
Excellent quality **handmade** 18 inch **doll** clothes for American Girl **dolls** and **...** We carry a variety of **handmade dolls**. Our line of **handmade dolls** include **...**
www.craftsfaironline.com/**Dolls**.html - Cached - Similar

Figure 5: The snippets (circled) of search engine result pages are leveraged for query expansion.

results are clicked on, as denoted in the indicator function $\mathcal{I}$ of the denominator of Z. Since the queries extracted from each searchable site usually contain some noisy queries that are not representative of the site content, by formulating the query language model for each searchable site with only clicked queries, we can reduce the amount of noisy queries significantly. In addition, our formulation of the site does not require direct sampling from the site itself as in [2], which can reduce tremendously the amount of computational effort. This site-level formula can also been maintained and updated per site basis, which is very flexible and can therefore updated in real time (will be discussed shortly).

For the third component in eq.(2), $\dfrac{1}{P(w)}$ is modeled as the inverse document frequency (idf) [18],

$$\frac{1}{P(w)} \propto \log \frac{N}{|\{q^* : w \in q^*\}|} \qquad (6)$$

where N is the total number queries in the training collection. As it is well-known, idf is able to assign higher weights to terms concentrated in a few documents of a collection rather than more general terms.

4.2.3 Optimize P(v) by Linear Feature Combination

Finally, we utilize the set of seven informative features that are collected for each searchable site to model the static rank of v, $P(v)$. The prior probability $P(v)$ is formulated as a weighted normalized linear combination of the seven collected features,

$$\begin{aligned} P(v) \quad = \quad & w_1 \frac{\log[\text{queries}(v)]}{\sum_{v' \in V} \log[\text{queries}(v')]} \\ & + \quad w_2 \frac{\log[\text{unique\_queries}(v)]}{\sum_{v' \in V} \log[\text{unique\_queries}(v)]} \\ & + \quad w_3 \frac{\text{clicks}(v)}{\sum_{v' \in V} \text{clicks}(v')} + w_4 \frac{\text{dt1}(v)}{\sum_{v' \in V} \text{dt1}(v')} \\ & + \quad w_5 \frac{\text{dt2}(v)}{\sum_{v' \in V} \text{dt2}(v')} + w_6 \frac{\text{Index}(v)}{\sum_{v' \in V} \text{Index}(v')} \\ & + \quad w_7 \frac{1/\text{entropy}(v)}{\sum_{v' \in V} 1/\text{entropy}(v')}. \end{aligned} \qquad (7)$$

In order to learn the optimal weight $[w_1, \ldots, w_7]$, we use the Simultaneous Stochastic Approximation (SPSA) algorithm proposed by [21, 20] as our gradient approximation method for non-convex objective function, since it is very efficient and requires only two objective function evaluations per iteration for gradient approximation. Here, we briefly review the SPSA algorithm (for the complete description of the algorithm referred to the work by Spall [21, 20]). In each iteration, a simultaneous perturbation vector $\Delta_k \in \mathcal{R}^d$ is generated according to the following conditions stated in [20]: Δ_k is a vector of d mutually independent mean-zero random variables $(\Delta_{k1}, \Delta_{k2}, \ldots, \Delta_{kd})$ satisfying $|\Delta_{kl}| \leq \alpha_0$ almost surely and $E|\Delta_{kl}^{-1}| \leq \alpha_1$ for some finite α_0 and α_1. As suggested in [21], each Δ_{kl} can be generated from the symmetrically Bernoulli distribution ($+1$ or -1 with equal probability). Given Δ_k, the gradient approximation is computed as,

$$\hat{g}_k(w_k) = \begin{bmatrix} 1/\Delta_{k1} \\ 1/\Delta_{k2} \\ \ldots \\ 1/\Delta_{kd} \end{bmatrix} \cdot \frac{L(w_k + c_k\Delta_k) - L(w_k - c_k\Delta_k)}{2c_k} \quad (8)$$

where $L(\cdot)$ is the objective function to be optimize. In our framework, we set the objective function $L(w)$ to be the accuracy performance of our system which is described in the experiment section. In addition, the parameters of the learning algorithm are set as the suggested values as in [21].

In summary, the task of searchable web site recommendation is formulated as a probabilistic model, $P(v|q)$. The conditional probability of a searchable web site v given a query q is computed in term of four components: (1) the query language model, $P(w|q)$; (2) the web site language model, $P(w|v)$; (3) the inverse document frequency, $\frac{1}{P(w)}$; and (4) the static rank, $P(v)$.

4.2.4 *Online Recommendation*

In Figure 6, we show that our framework of searchable web site recommendation can be incorporated into an online setting. Given the current client browser log data, the system extracts potential searchable web sites and submitted queries which then are used to construct the language models for the queries and web sites. When users issue new queries, the system proposes a ranked list of suitable searchable web sites. At this point, new client browser log data can be collected and use as feedback to update our language models. This continuous effort can further prune and refine the model on a daily basis, without any human involvement.

5. EXPERIMENT

In this section, we empirically analyze the performance of the proposed framework. We first introduce the performance metrics. We then describe in details our experimental setup. Finally, we present the empirical results.

5.1 Experiment Setup and Result

Due to the novelty of the problem of recommending searchable web sites, there is no existing benchmark test data for this task. Ideally, in order to evaluate the performance of our system, we would need a ranked list of best matched searchable web sites for each test query. However, the benchmark data would require time-consuming human-labeling effort.

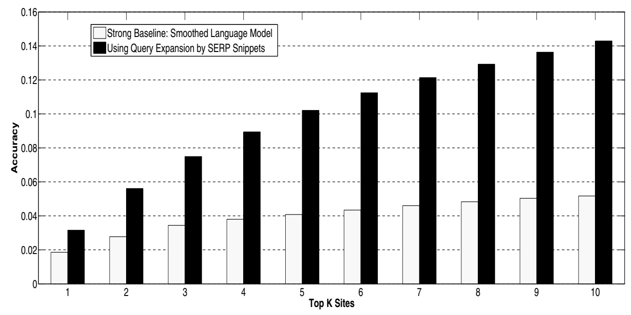


Figure 7: Accuracy of top-K sites with $K = 1$ to 10. Using query expansion increases performance significantly than using query string itself [22].

In our experiment, we first extracted all queries issued directly to the web sites during a two-week period in the client side log. We use the first week of the log for training our system and the second week for testing. We select those *novel* test queries from the second week that have never appeared in the training set. Given a set of queries and the sites that they were issued to, $\text{QV} = \{(q_1, v_1), (q_2, v_2), \ldots, (q_n, v_n)\}$, the accuracy of the system with respect to the top K recommended sites is computed as

$$\text{Accuracy}(\text{QV}, K) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{I}_K(q_i, v_i), \quad (9)$$

where

$$\mathcal{I}(q, v) = \begin{cases} 1 & v \in \text{TopVertical}_K(q) \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where $\text{TopVertical}_K(q)$ is the set top K verticals returned by the ranking system. For simplicity, we model the query and the web site language model with uni-gram vocabulary. In our experiment, we first compare the performance of query expansion using search result snippets with query string itself.

Because of the novelty of our data source and feature construction method, it is unable to compare with existing supervised methods [2] or methods that require external knowledge [13]. A fair comparison should be made to unsupervised language models. Consequently, we built a strong baseline approach which is a smoothed language model reported as the best model in [22]. In Figure 7, we plot the accuracy performance of our system with and without the use of query expansion, with K from 1 to 10. As the figure clearly demonstrates, query expansion with snippets gives significantly better performance across different top recommended searchable web sites. On average, the query expansion improves the accuracy by 240%.

In the second series of experiments, we demonstrate how different weighting scheme for the prior probability $P(v)$ effects the overall performance of our recommendation framework.

5.1.1 *Constant Prior vs. Uniform Weighting*

In Figure 8, we investigate the constant prior and the uniform weighting scheme. In the constant prior scheme, we set $P(v) \propto 1$, meaning that the prior probability is the same for all sites. While in the uniform weighting, we set $w_1 = \cdots = w_7 = 1$, so that each of the seven features

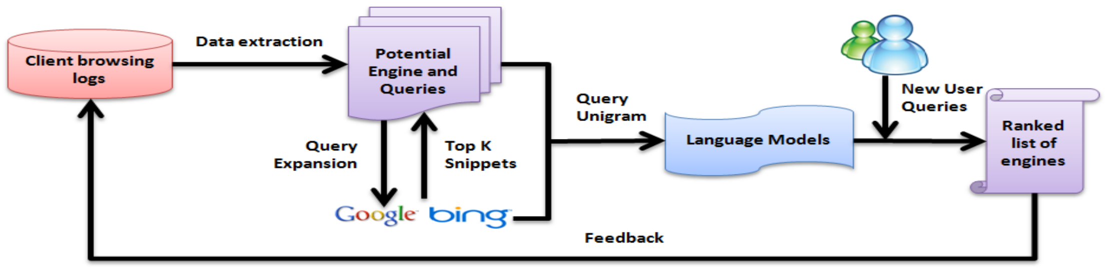


Figure 6: The framework of online searchable web site recommendation.

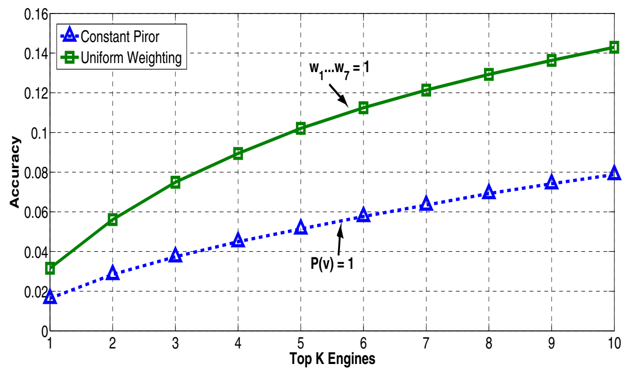


Figure 8: Constant prior vs. uniform feature weighting. Result indicates that the features extracted for $P(v)$ is performs significantly better than constant prior for all searchable web sites.

are treated equally. However, $P(v)$ could be different for different sites in this case. We observe that our system using uniform weighting performs significantly better than the one with constant prior. The conclusion from this experiment is that the extracted features for each searchable web site provide discriminative information to our ranking system, which is better than simply treating all searchable sites with equal importance.

5.1.2 *Individual Feature Comparison*

Now we investigate how much each of these features contributes to the overall performance of our recommendation system. In Figure 9, we plot the performance of different weighting schemes for the static ranking which assign $w_i = 1$ for the i^{th} feature and $w_j = 0$ $(j \neq i)$ for all other features as in eq.(2). We observe that the features are clustered into 3 groups. First, we observe that static rankings which utilize number of queries, number of unique queries, or number of clicks all produce similar results to each other and outperform the uniform weighting scheme. We also observe that static rankings utilizing dwell-times on the search result page or the referred pages produce similar results to uniform weighting scheme. Finally, static ranking utilizing entropy or Index do not make significant difference compared to constant weighting scheme. Hence, we can order the significance of the features as following: number of queries, number of unique queries, number of clicks, dwell-time on the search results, dwell-time on the referred pages, entropy and pages indexed by search engines.

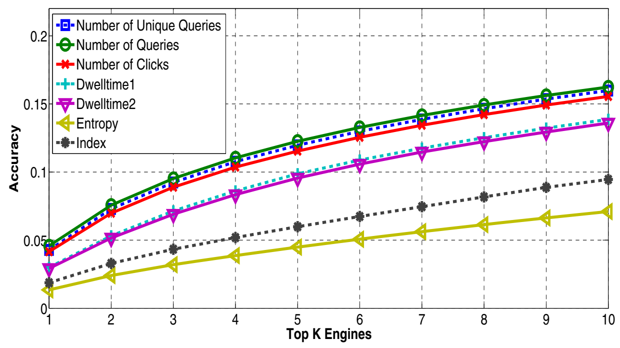


Figure 9: Accuracy comparison of each individual features by setting $w_i = 1$ and the rest to be 0. Number of unique queries is the best feature for prediction.

Table 5: The optimal feature weights learnt from SPSA method. The weights do not sum up to 1.

Feature	Optimal Weight
Number of queries	0.826578359
Number of unique queries	0.172329984
Number of clicks	0.133643353
Dwelltime 1	0.105575731
Dwelltime 2	0.073425058
Index	0.100119786
Entropy	0.051705972

5.1.3 *Optimized Feature Combination*

Finally, in Table 6 we list the performance improvement of the system using the weights trained by the SPSA algorithm. The improvement is calculated by extracting the best of three (Constant Prior, Uniform Weight and # of Unique Queries) from the optimal combination. We observe that the learning algorithm SPSA is able to learn a weight combination that outperforms other weight schemes significantly. The optimal weights for the seven features are listed in Table 5, the order of which is consistent with what we observed in Figure 9. Note that since SPSA does not put normalization constraints on the weights, the summation of the seven weights is not equal to one. In practice, it does not affect the static rank $P(v)$ as long as the relative order of the prior probability is preserved, so that more popular sites are more likely to be selected.

Table 6: Performance improvement of our feature optimization algorithm against other results. The improvement is statistically significant (p-value < 0.001).

K	Constant Prior	Uniform Weight	# of Unique Queries	Optimal combination	Improvement (%)
1	0.016375719	0.031560226	0.04579501	**0.046729602**	+0.0934592
2	0.02844877	0.05609423	0.075644265	**0.077492018**	+0.1847753
3	0.037340215	0.074910143	0.095222939	**0.097702723**	+0.2479784
4	0.044964793	0.089394504	0.110397625	**0.113840474**	+0.3442849
5	0.051380278	0.102083795	0.122546001	**0.126965808**	+0.4419807
6	0.057649957	0.112433304	0.132723569	**0.138783011**	+0.6059442
7	0.063418942	0.12135501	0.141539634	**0.145944035**	+0.4404401
8	0.069234695	0.129256071	0.149226056	**0.155809357**	+0.6583301
9	0.074160478	0.136326309	0.156173763	**0.166392706**	+1.0218943
10	0.078797399	0.142921989	0.162401625	**0.168995215**	+0.6593590

6. DISCUSSION ON QUERY EXPANSION

The most effective technique we proposed in this paper is automatic query expansion using search engine result snippets, which leads to approximately 5% accuracy gain against constant prior with $P(v) = 1$ for all sites. In order to show the necessity of query expansion, we examine the performance of a simplified vector matching algorithm in predicting queries at 10 searchable web sites with high query volume in our data. Specifically, we measure whether queries issued to a web site in our test data match queries from the training data in terms of cosine similarity, without using any query expansion techniques.

For a searchable web site v, we obtain the sets U_v of all unigrams and B_v of all bigrams in queries issued to v in our training data, along with frequencies $f_{x,v}$, where $f_{x,v}$ denotes the number of occurrences of $x \in U_v \cup B_v$ among queries submitted to v. For $x \notin U_v \cup B_v$, we set $f_{x,v} = 0$. In addition, we collect identical information for a general purpose search engine s. We denote the associated unigrams and bigrams as U_s and B_s, and denote the associated frequencies as $f_{x,s}$ for $x \in U_s \cup B_s$.

For a query q with distinct tokens $w_0, \cdots, w_n$, with frequencies $|w_0|, \cdots, |w_n|$ (where typically every $|w_i| = 1$), we compute the similarity between the query q and the vertical v as

$$\cos(q, v) = \frac{\sum_{i=1}^{n} |w_i| f_{w_i,v}}{\sqrt{\sum_{i=1}^{n} |w_i|^2 \sum_{x \in U_v \cup B_v} f_{x,v}^2}} \qquad (11)$$

This cosine metric provides a relative ranking of searchable web sites suitable for each query. Whenever $\cos(q, v)$ is sufficiently greater than $\cos(q, s)$, we treat v as a likely match to the user's query. Note that this is a canonical document vector retrieval algorithm based on cosine similarity (c.f. [3]) where, rather than representing a searchable web site v by the text on the site itself, v is represented by a virtual document constructed from all queries issued to v in the training data.

To measure the performance, we determine which of the 10 high-volume web sites are predicted for each query in the test data. If a website is predicted for the query and the query occurs on that site in the test data, it is a true positive (TP). If the web site is predicted for the query and the site does not occur for the query, it is a false positive (FP). If the web site is not predicted for the query, but it occurs for the query in the test data, it is a false negative (FN). Precision is computed on a site-by-site basis as TP/(TP + FP) and recall is computed as TP/(TP + FN). F1 is computed as the harmonic mean of precision and recall.

Table 7 summarizes the performance for this algorithm, where v is predicted for q whenever $\cos(q, v) > 20 \cos(q, s)$. You Tube performs best in terms of precision, given that its query volume in the training data is much larger than the other 9 sites. However, the F1 scores are poor across all 10 web sites, with no F1 score exceeding 50%, and several falling below 10%. This result indicates that even for popular searchable web sites, in the absence of large amounts of data, proper query expansion, or smoothing methods, performance is poor.

7. CONCLUSION AND FUTURE WORK

In this paper, we proposed a new methodology of discovering and recommending searchable web sites from client-side browsing logs. Our framework utilized automatically extracted features from the logs to construct a unigram language model for site recommendation. We proposed an efficient query expansion technique to address the query sparseness issue by leveraging the search engine result snippets. Experimental results indicated that with the language model and query expansion, our framework can significantly improve the recommendation performance than a simple constant prior Bayesian method.

Comparing with previous approaches, our method was the first-of-a-kind in that (1) the proposal was completely unsupervised, making it capable of handling terabytes of data, (2) the approach focused on site-level recommendation rather than topic-level query categorization [2], so that the features extracted from each site can dynamically adjust at site-level rather than making dramatic change by adding or removing categories, (3) the features we used from click-site logs respect the user perspective of site popularity and thus can potentially discover valuable sites in which users perform significant amount of searches, even ahead of the awareness of general search engines.

There are a lot of future directions. Firstly, we have used client-side log (queries issued to the web sites directly) to train and evaluate our recommendation system in an offline experiment. Since our intended use for this system is to complement the current search engine's web page results and ad results, it would be valuable to run online experiments where the users with information need are presented with the searchable web site recommendations embedded in the search engine result pages. These experiments will enable us to evaluate how much value our system adds to

Table 7: Performance of a simplified matching algorithm for 10 high-volume web sites. Without query expansion or smoothing methods, performance is quite poor.

Web Site	TP	FP	TN	FN	Precision	Recall	F1 Score
amazon	1817	8644	47000	2294147	0.173693	0.037221	0.061304
etsy	4406	7152	11912	2328138	0.381208	0.270009	0.316114
flickr	1640	8820	35146	2306002	0.156788	0.044582	0.069424
hulu	3034	26849	18609	2303116	0.101529	0.140184	0.117766
imdb	3114	17158	36454	2294882	0.153611	0.0787	0.104078
indeed	10379	1390	21296	2318543	0.881893	0.327672	0.477811
myxer	3317	54719	23873	2269699	0.057154	0.121993	0.07784
target	4640	9034	29954	2307980	0.33933	0.134127	0.19226
walmart	4548	7061	51847	2288152	0.391765	0.080645	0.133757
youtube	80558	815	2058546	211689	0.989984	0.03766	0.072559

the users and how the recommendations interact with other types of search results. Secondly, many searchable web sites serve a very niche domain. Incorporating a user's contextual information, such as location, query history and time of day, would allow us to recommend the web sites much more accurately. For example, if the user queries a book title, a link to search his local community library would be a nice complement to the usual web results. Thirdly, in the current system, we assume that the language models of the queries issued to the general search engine and to the web site directly are the same. A user may issue "britney spears video" to a general search engine but simply "britney spears" to Youtube. A study of the language model difference would allow us transform the user query to a more appropriate site-specific query. Finally, our searchable web site discovery algorithm only handles keyword-based search sites. Many sites offer structured search. For example, Southwest Airline's web site allows the user to search flights by different options. Understanding the structured search URL automatically and transforming the user's keyword-based query to a structured site-specific search URL would be a promising future direction.

8. REFERENCES

[1] Netscape communication corporation. open directory project. http://www.dmoz.org.

[2] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo. Sources of evidence for vertical selection. In *The 32nd Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR09)*, 2009.

[3] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[4] L. Barbosa and J. Freire. Combining classifiers to identify online databases. In *WWW '07*, pages 431–440, 2007.

[5] S. M. Beitzel, E. C. Jensen, D. D. Lewis, A. Chowdhury, and O. Frieder. Automatic classification of web queries using very large unlabeled query logs. *ACM Trans. Inf. Syst.*, 25(2):9, 2007.

[6] S. K. Bhavnani. Domain-specific search strategies for the effective retrieval of healthcare and shopping information. In *Conference on Human Factors in Computing Systems*, pages 610–611, 2002.

[7] D. J. Brenes, D. Gayo-Avello, and K. Perez-Gonzalez. Survey and evaluation of query intent detection methods. In *The Workshop on Web Search Click Data (WSCD09)*, 2009.

[8] W. Chang, P. Pantel, A.-M. Popescu, and E. Gabrilovich. Towards intent-driven bidterm suggestion. In *WWW '09*, pages 1093–1094, 2009.

[9] J. Cope, N. Craswell, and D. Hawking. Automated discovery of search interfaces on the web. In *ADC '03*, pages 181–189, 2003.

[10] F. Diaz. Integration of news content into web results. In *WSDM '09*, pages 182–191, New York, NY, USA, 2009. ACM.

[11] F. Diaz and J. Arguello. Adaptation of offline vertical selection predictions in the presence of user feedback. In *The 32nd Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR09)*, 2009.

[12] E. Diemert and G. Vandelle. Unsupervised query categorization using automatically-built concept graphs. In *The 19th International World Wide Web Conference(WWW09)*, pages 461–470, 2009.

[13] J. Hu, G. Wang, F. Lochovsky, J.-T. Sun, and Z. Chen. Understanding user's query intent with wikipedia. In *The 19th International World Wide Web Conference(WWW09)*, pages 471–480, 2009.

[14] B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing & Management*, 36(2):207–227, March 2000.

[15] X. Li, Y.-Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *SIGIR08*, pages 339–346, 2008.

[16] D. J. C. Mackay and L. C. B. Peto. A hierarchical dirichlet language model. *Natural Language Engineering*, 1(1):289–307, 1995.

[17] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's deep web crawl. *Proceedings of the VLDB Endowment*, 1(2):1241–1252, 2008.

[18] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Managment*, 24(5):513–523, 1988.

[19] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *SIGIR '06*, pages 131–138. ACM, 2006.

[20] J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37:332–341, 1992.

[21] J. C. Spall. Implementation of the simultaneous perturbation algorithm for stochastic approximation. *IEEE Transactions on Aerospace and Electronic Systems*, 34:817–823, 1998.

[22] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, April 2004.