Actively Predicting Diverse Search Intent from User Browsing Behaviors*

Zhicong Cheng
School of Software and Microelectronics
Peking University
Beijing, 100871, P. R. China
czc0316@live.com

Bin Gao
Microsoft Research Asia
4F, Sigma Center, No. 49, Zhichun Road
Beijing, 100190, P. R. China
bingao@microsoft.com

Tie-Yan Liu
Microsoft Research Asia
4F, Sigma Center, No. 49, Zhichun Road
Beijing, 100190, P. R. China
tyliu@microsoft.com

ABSTRACT

This paper is concerned with actively predicting search intent from user browsing behavior data. In recent years, great attention has been paid to predicting user search intent. However, the prediction was mostly passive because it was performed only after users submitted their queries to search engines. It is not considered why users issued these queries, and what triggered their information needs. According to our study, many information needs of users were actually triggered by what they have browsed. That is, after reading a page, if a user found something interesting or unclear, he/she might have the intent to obtain further information and accordingly formulate a search query. Actively predicting such search intent can benefit both search engines and their users. In this paper, we propose a series of technologies to fulfill this task. First, we extract all the queries that users issued after reading a given page from user browsing behavior data. Second, we learn a model to effectively rank these queries according to their likelihoods of being triggered by the page. Third, since search intents can be quite diverse even if triggered by the same page, we propose an optimization algorithm to diversify the ranked list of queries obtained in the second step, and then suggest the list to users. We have tested our approach on large-scale user browsing behavior data obtained from a commercial search engine. The experimental results have shown that our approach can predict meaningful queries for a given page, and the search performance for these queries can be significantly improved by using the triggering page as contextual information.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

SearchTrigger, search intent, diversification, log mining, contextual information

*This work was performed when the first author was an intern at Microsoft Research Asia.

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

1. INTRODUCTION

User intent understanding has become a hot topic in Web search and data mining, and commercial search engines have realized its importance for providing better search experiences. For example, Google, Yahoo!, Bing[1], and Ask all provide *query suggestion* [1, 3, 13] as a prediction of user search intent. While such a technique has achieved certain success, it also has obvious limitations, some of which are listed as below.

- The prediction was conducted in a passive manner, in the sense that it was performed only after users submitted their queries to search engines. Note that Web users spend most of their time not on search, but instead on browsing, authoring, and so on. Therefore in most cases, when users have *latent* search intents, search engines cannot make meaningful predictions. In this regard, the impact of passive intent prediction on Web users will not be sufficiently significant.

- The prediction was usually based on historical queries issued by the user, or similar queries issued by other users. Search engines are not aware how the information need of a user was originally generated and what motivated him/her to issue the query. As a result, the contextual information that search engines can obtain will be insufficient to produce high-quality and personalized search results.

In order to overcome the aforementioned limitations, it is desired to predict users' search intents in an active manner, based on their behaviors even beyond search (e.g., browsing, authoring, etc.), and leverage information related to these behaviors to improve search quality. This is exactly the motivation of our work.

For this purpose, first of all, we should understand the factors that may trigger users' information needs. According to our study, in a significant proportion of cases, information needs are generated when users browse the Web. Our analysis (see Section 2) on user browsing behavior data shows that about 19.3% browsing sessions contain "*browse* → *search*" patterns (i.e., the user searched something right after he/she browsed a page). In 66% of such sessions, some search queries were almost certainly triggered by the contents of the pages that users browsed before their search actions. That is, when browsing a page, the user might find something interesting or unclear in the page, or be reminded

[1]The new generation of Windows Live Search.

of something related to the page. Then he/she might want to conduct a search immediately after browsing the page, to learn more about all of this. For example, Linda is browsing the news of Michael Jackson's death and she wants to listen to some songs of Jackson to honor the memory of him. So she comes up with the search query *Music of Michael Jackson*. For ease of reference, we call this scenario *SearchTrigger*. More accurately, *SearchTrigger* refers to a "*browse* → *search*" pattern, in which the search query is triggered by the content of the browsed page. We call the query a *SearchTrigger* query, and the pattern a *SearchTrigger* pattern. Our further analysis shows that the *SearchTrigger* queries are very diverse. For 92% pages, the corresponding queries fall into at least two dissimilar topical clusters.

A real example of *SearchTrigger* that we found during the study is given as follows.

Example 1. (*SearchTrigger.*) A user is a fan of Shakespeare. He/she opened a webpage about Shakespeare's FAQ (`http://absoluteshakespeare.com/trivia/faq/faq.htm`). There are tens of FAQs in the page about Shakespeare. For instance, there is a question "*Is it true nobody knows Shakespeare's birthday?*" at the top of the page. The answer is: "*It is true we don't know Shakespeare's date of birth. We know it was in 1564 but our only record at this time was of his baptism at the Holy Trinity Church on April the 26th. By convention and some guesswork, Shakespeare's birthday is by tradition celebrated three days earlier on April the 23rd.*" After reading the notes, the user felt interested in the birthday of Shakespeare. To satisfy his/her curiosity, the user raised a query "*when was Shakespeare born*" to a search engine for more answers.

Note that the *SearchTrigger* queries are those queries triggered by the browsed page, but usually NOT the key phrases of the page or the queries that have the page as its top search result. For instance, in the above example, "*when was Shakespeare born*" does not appear in the page of Shakespeare's FAQ, and therefore is not one of the key phrases. Also if we search "*when was Shakespeare born*" using major search engines, the page of Shakespeare's FAQ even does not appear in the top-100 results. This is actually reasonable. If the query is a key phrase of the page or the page is in the top search results for the query, it is very likely that the page has already contained the answer to the query and it is unnecessary for the user to issue the query to search engines. The query is submitted usually because the user wants to obtain some novel information that is not covered by the triggering page.

If we can predict the *latent* search intent in the *SearchTrigger* scenario, and suggest meaningful queries to users when they are browsing, we will be able to help both search engines and their users. On one hand, by providing search shortcuts corresponding to these suggested queries, we actually extend the search function to outside the search box, and provide more opportunities for users to use search engines. On the other hand, by using the historical browsing behaviors (including the content of the pages browsed by the user) as contextual information, we can improve search accuracy for these suggested queries and provide much better user experiences. This task is, however, non-trivial due to the following reasons: (i) not all "*browse* → *search*" patterns correspond to real *SearchTrigger*; (ii) the search intents of users can be very diverse even if they read the same page, according to the statistics given in Section 2.

To tackle the challenging task, in this work, we propose a series of technologies.

1. Given a page, we extract all the queries that users searched right after reading the page from user browsing behavior data.

2. We learn a ranker to effectively sort these queries according to their likelihoods of being *SearchTrigger* queries.

3. We propose an optimization framework to diversify the ranked list of queries obtained in the second step, and present the diversified ranked list of queries to users.

We have tested our proposed approach on large-scale user browsing behavior data, and develop a contextual retrieval algorithm to leverage the page that triggers a query to improve search accuracy for the query. The experimental results have shown that the approach can improve user experience and enhance search accuracy.

To sum up, the contributions of this work are as below.

- We have proposed the concept of active prediction of users' search intents, which extends the functionality of search engines beyond their original boundaries.

- By mining user browsing behavior data, we have discovered a special pattern called *SearchTrigger*, in which the search intent is triggered by the page that a user visits right before his/her search action.

- We have found that *SearchTrigger* queries for the same page are usually very diverse. Accordingly, we have proposed a method to suggest a diversified query list for a given page and demonstrated its effectiveness through a contextual retrieval algorithm.

The rest of the paper is organized as follows. In Section 2, we present the study on user browsing behavior data, and show that *SearchTrigger* is a popular pattern in such data. The algorithms to effectively predict *SearchTrigger* queries for a given page are introduced in Section 3. Experimental results are discussed in Section 4. Conclusions and future work are presented in the last section.

2. ANALYSIS ON USER BEHAVIOR DATA

In order to better understand how what users browsed triggered their search intents, we have conducted an extensive study on user browsing behavior data, as reported in this section.

The primary source of data for this study was the anonymous logs of URLs visited by users who opted in to provide data through a widely-distributed toolbar on Internet browsers. Each log entry is a tuple of {*user ID*, *timestamp*, *URL*}, meaning a user visited a URL at some time. *User ID* was encrypted by an irreversible hash function. Intranet and secure (e.g., *https*) URL visits were all removed from the source. To minimize the influence caused by linguistic and regional variations, we only kept the records generated in the United States. As a result, the data consist of about 3 billion anonymous page views in 32 successive days during May and June in 2007.

2.1 "Browse → Search" Pattern Extraction

We extracted "*browse → search*" patterns from all sessions in the user browsing behavior data. Here we define a session as a logical unit of time-ordered user browsing activities. For each user's data, we start a new session if there is more than 30 minutes of inactivity between the current page view event and its immediate preceding event [17]. For the page view events and sessions, we have the following definitions.

Definition 1. **Search Portal Event / Search Event / Browse Event.** If a page view event contains the URL of a (general or vertical) search engine portal[2] but does not contain any search query, we call the event a search portal event; if the page view event contains the URL of a search engine and a query as its argument, we call the event a search event; otherwise, we call the event a browse event.

For example, in the following session, the first URL corresponds to a browse event, the second a search portal event, and the third a search event.

Example 2. (*Three successive URLs in a session.*)
`http://www.apple.com/iphone/`
`http://www.google.com/`
`http://www.google.com/search?q=iphone+release+date`

Definition 2. **Search Session / Non-search Session.** If a session contains at least one search event, it is called a search session. Otherwise, it is a non-search session.

Definition 3. **"Browse → Search" Pattern.** After excluding all search portal events from a search session, if there is a search event immediately after a browse event, we call the tuple {*URL*, *query*} a "browse → search" pattern where *URL* is the page visited in the browse event and *query* is extracted from the search event.

For instance, from Example 2, one can extract the following "browse → search" pattern, {`http://www.apple.com/iphone/`, *iphone release date*}. Note that there is only one case that a search session does not contain any "browse → search" pattern: all search events happened prior to the browse events in the session. In this case, this is no evidence that the search events are triggered by the content of previously visited pages. Therefore, they are not in the interested scope of our study.

Definition 4. **"Browse → Search" Session.** If a search session contains at least one "browse → search" pattern, it is called a "browse → search" session.

We processed the entire user browsing behavior data in our study, and obtained the statistics as shown in Table 1. From the table, we can see that about 19.3% sessions are "browse → search" sessions and the average number of "browse → search" patterns per such session is 5.6. As will be seen in the next subsection, these "browse → search" patterns potentially correspond to *SearchTrigger*, but not necessarily all qualified *SearchTrigger*.

[2]Note that in our study, we only included the search events from Google, Yahoo!, Windows Live Search, AOL, and Ask, because the majority of the search market share in the United States (around 90% according to [24, 25, 26]) comes from these five search engines.

Table 1: Statistics on sessions and "browse → search" patterns.

Entity	Quantity
Log entries	2,998,754,253
Unique URLs	940,555,664
Sessions	153,663,449
Non-search sessions	116,399,857
Search sessions	37,263,592
"Browse → search" sessions	29,695,191
"Browse → search" patterns	167,570,019
Average entries per session	19.5
Percentage of non-search sessions	75.7%
Percentage of search sessions	24.3%
Percentage of "browse → search" sessions	19.3%
Average "browse → search" patterns per "browse → search" session	5.6

2.2 Identification of SearchTrigger Queries

We randomly sampled 200 "browse → search" sessions and asked experienced human analysts to perform further analysis on the data. As a result, we found 849 "browse → search" patterns, the queries in which can be classified into seven categories.

1. Key phrase of the page
The query is a key phrase of the browsed page. For example, after visiting a page about Shakespeare's FAQ (`http://absoluteshakespeare.com/trivia/faq/faq.htm`), a user issued the query *Shakespeare's play*, which is a key phrase in the page due to its high relevance with the main topic of the page and its high frequency.

2. Information in the page but not key phrases
The query describes some interesting part of the browsed page, but it is not a key phrase. The query given in Example 1 belongs to this category. Furthermore, information in the page such as images and flashes may also trigger queries belonging to this category. We sent the queries in this category to major search engines, and found that their corresponding top-10 search results do not contain the browsed pages.

3. Famous site
Many users have the experiences that they wanted to visit a famous website such as *facebook, youtube, myspaces*, but they forgot its exact URL. They then issued a query like *facebook* to a search engine to get a shortcut to the portal of the website. Such queries are usually not triggered by the page that a user previously browsed.

4. Unrelated topic
Sometimes, a user searched a query absolutely unrelated to the content of the previous page. There might be various reasons for this situation. For example, the user changed his/her interest to another totally different topic, or it is some information need from his/her daily life that triggered the query (e.g., the user felt hungry and searched for the phone number of a restaurant).

5. Repeated search
Sometimes a user first submitted a query to a search engine, and browsed a page in the search results. How-

Table 2: Statistics of "browse → search" patterns.

Category		Number of Queries		Percentage		Average Pattern Frequency
SearchTrigger	Key phrase of the page	202	23	23.8%	2.7%	26.8
	Information in the page but not key phrases		179		21.1%	2.5
Non-SearchTrigger	Famous site	604	15	71.1%	1.8%	92.9
	Unrelated topic		121		14.2%	1.6
	Repeated search		350		41.2%	52.0
	Query refinement		118		13.9%	1.8
Cannot judge	Cannot judge	43		5.1%		1.0

ever, he/she was not satisfied with the page. Then he/she clicked the back button in the browser to try another page in the search results, or changed to another search engine and searched the query again. As a result, we can extract a sequence of events like $query \rightarrow URL_1 \rightarrow query \rightarrow URL_2$. Here the second *query* is not triggered by the content of URL_1 since it is simply a repeated search.

6. Query refinement

Sometimes a user first submitted a query to a search engine, and browsed a page in the search results. However, he/she was not satisfied with the page and he/she refined the query and resubmitted it to the search engine. As a result, we can obverse the following sequence of events, $query_1 \rightarrow URL_1 \rightarrow query_2 \rightarrow URL_2$. In this case, $query_2$ is not triggered by the content of URL_1 either.

7. Cannot judge

It was difficult for the analysts to categorize all the "browse → search" patterns accurately. In some cases, the analysts could not make sure which category a pattern should belong to. In some other cases, the page in a pattern has been expired or needs login (e.g., forums) to view its content. We regard all these cases as *cannot judge*.

The statistics of the above categories are summarized in Table 2. In addition to the number of queries and percentage in the 849 patterns, we also count the frequency of a "browse → search" pattern in the entire user browsing behavior data (see the last column in Table 2), which can reflect whether a particular pattern is popular or not.

According to the definition of *SearchTrigger* given in the introduction, the human analysts thought that both the first and the second categories in the study correspond to *SearchTrigger*, categories 3 to 6 are not *SearchTrigger*, and category 7 is not judgeable. From this result, we can come to the following conclusions.

- About a quarter (23.8%) of "browse → search" patterns contain *SearchTrigger* queries. Further analysis shows that 132 of the sampled 200 sessions (66%) contain at least one *SearchTrigger* query. All these numbers show that *SearchTrigger* is a frequent pattern of user behaviors and it is worthy of further investigation.
- Among the *SearchTrigger* queries, the proportion of queries belonging to category 2 is significantly larger than that belonging to category 1. This coincides with our discussions in the introduction: most *SearchTrigger* queries (in our study, 88.6%) are not key phrases of the browsed page.

Table 3: Statistics on diverse *SearchTrigger* queries.

Number of Search Intents	Number of Pages
≤ 1	8
$2 \sim 5$	12
$6 \sim 9$	68
≥ 10	12

- According to the average pattern frequency in each category, we can see that not all high-frequency patterns correspond to *SearchTrigger* (e.g., famous site and repeated search also have very high frequencies).

2.3 Diversity in SearchTrigger Queries

We randomly sampled 100 pages from the user browsing behavior data. Then we collected the queries in all the "browse → search" patterns containing the page. We asked human analysts to judge whether these queries are *SearchTrigger* queries. For the queries that are judged as *SearchTrigger* queries, human analysts further grouped them into several clusters according to their corresponding search intents. For example, after reading a page about rabbits (`http://exoticpets.about.com/od/rabbits/Rabbits.htm`), users issued 11 different queries, 7 of which were identified as *SearchTrigger* queries. These queries were organized into three groups, i.e., {*rabbits, pet rabbit, wild rabbits*}, {*rabbits pictures, pet pictures*}, and {*rabbit care guide, rabbits breeds*}, indicating three different search intents. The statistics of this study are summarized in Table 3. From the table we can see that about 92% pages triggered at least two different search intents, showing that *SearchTrigger* queries are often diverse. Our explanation to this observation is that a page may contain several different (but correlated) topics and each of them can motivate users to search something.

3. PREDICTING DIVERSE SEARCH INTENT

The statistics in Section 2 show that *SearchTrigger* is a frequent pattern of user behaviors. Then if search engines can accurately predict users' intents ahead of time and suggest *SearchTrigger* queries while users are browsing a page, they can provide timely search function when users need it, and thus greatly improve user experiences. This task is, however, non-trivial, as discussed in the introduction. To tackle the challenges, we propose a set of techniques. First, we extract the "browse → search" patterns from user browsing behavior data, and build a candidate query set for each page. Then, given a page and its candidate queries, we extract various features and learn a ranking model to sort these queries according to their likelihoods of being *SearchTrigger*

queries. After that, we adopt an optimization algorithm to diverse the ranked list of the *SearchTrigger* queries. This diversified query list will be presented to users as suggestions.

3.1 Problem Definition

Given a page, the task is to predict a ranked list of *SearchTrigger* queries that a random user may want to issue after reading the page, based on historical user browsing behavior data.

To this end, one can segment user browsing behavior data into sessions, and extract all "browse → search" patterns. For each page p, a candidate query set can be generated by aggregating all its co-occurrence queries in the patterns. Suppose the candidate query set is $S_p = \{q_1^{(p)}, q_2^{(p)}, \cdots, q_m^{(p)}\}$. Then a straightforward solution to the task is to count the frequency of each query in S_p, and suggest the most frequently-asked queries to users. However, this naive method would not work well, because 71.1% candidate queries are non-SearchTrigger, and many of them are of high-frequencies, according to the analysis in Section 2.

To solve the aforementioned problem, we propose extracting multiple features for each candidate query and adopting machine learning technologies to rank these candidate queries according to their likelihoods of being *SearchTrigger* queries based on the features.

3.2 Query Features

On one hand, the task of query ranking can be regarded as a dual problem of document ranking in search. Therefore, it is straightforward to also extract query-document matching features for the task, which are widely used in the literature of document ranking. For example, we extract the following features to describe the matching between a query and its preceding page: term frequency (TF)[2], inverse document frequency (IDF)[2], TF *IDF[2], LMIR with ABS smoothing (LMIR.ABS)[20], LMIR with DIR smoothing (LMIR.DIR)[20], and LMIR with JM smoothing (LMIR. JM)[20]. If we consider that each page contains three parts, i.e., *url*, *title*, and *body*, we will have 18 query-document matching features in total. In addition, we also extract the following features: *length of query*, *unique word count of query*, and *maximum word length of query*. In many cases, we had better not suggest long queries (or query words) to users, because most of them are rarely asked by real users. These features can help avoid such cases in the suggested queries.

On the other hand, however, there is also difference between the task of query ranking and that of document ranking. In the former case, each candidate query is represented by features while in the latter case each candidate document is represented by a set of features. This difference actually poses a challenge to us: queries are usually much shorter than documents, which makes the above content matching features not informative enough to describe queries. For example, only from the query word, it is difficult to judge whether a query reflects the interesting part of a page that can attract users' attention. Such information, which is important for identifying *SearchTrigger* queries, need to be extracted from other information sources, e.g., the bipartite graph as described below.

In our work, we extract all "browse → search" patterns from user browsing behavior data, and build a page-query bipartite graph. In this bipartite graph, a *page node* is created for each unique page, while a *query node* is created for each unique query in the patterns. An edge e_{ij} is generated between page p_i and query q_j if they co-occur in a "browse → search" pattern. The weight w_{ij} of edge e_{ij} is the frequency of such patterns. An example page-query bipartite graph is shown in Figure 1. From the bipartite graph, we extract the following features, in hope to describe users' interests:

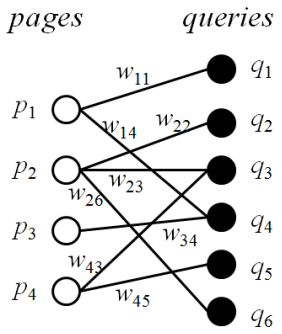

Figure 1: An example of page-query bipartite graph.

- *Query Visibility.* We call the number of edges linking to a query *query visibility*. If a query has large query visibility, it means that users would ask the query after visiting many different pages.
- *Query Popularity.* We call the sum of weights of all the edges linking to a query *query popularity*. If a query has large query popularity, its total number of occurrences in the extracted patterns is large.
- *Pattern Frequency.* We call the weight of the edge between a query and the given page *pattern frequency*. This feature reflects whether the same query is issued by many different people after reading the page.

3.3 Learning to Rank Candidate Queries

Previous work [4, 9, 12] has shown the advantage of using a learning to rank approach over using heuristic rules, especially when there are multiple evidences of ranking to be considered. Given the query features as described in Section 3.2, we also adopt a learning to rank technique to rank the candidate queries.

Given page p and its candidate query set $S_p = \{q_1^{(p)}, q_2^{(p)}, \cdots, q_m^{(p)}\}$, where m is the number of queries. Let $X \subset R^d$ be the feature space of queries, where d is the number of features. Then $x_i^{(p)} \in X$ denotes the feature vector of $q_i^{(p)}$ with respect to p, $i = 1, 2, \cdots, m$. Suppose $Y = \{l_1, l_2, \cdots, l_K\}$ is the set of labels representing the likelihood that a query is a *SearchTrigger* query for the document. Assume that there is a total order between the labels, i.e., $l_1 > l_2 > \cdots > l_K$. In our study, K is set to 3, and l_1, l_2, and l_3 represent the labels of *SearchTrigger*, *cannot judge*, and *non-SearchTrigger* respectively.

In the training process, there is a set of n pages, their candidate queries, and the corresponding labels (given by human annotators), i.e., $Z = \{z_1, z_2, \cdots, z_p, \cdots, z_n\}$, in which $z_p = \{(x_1^{(p)}, y_1^{(p)}), (x_2^{(p)}, y_2^{(p)}), \cdots, (x_m^{(p)}, y_m^{(p)})\}$. Here $x_i^{(p)} \in X$ is the feature vector of $q_i^{(p)}$ and $y_i^{(p)} \in Y$ is its label. If we have $y_i^{(p)} > y_j^{(p)}$, then we can say $q_i^{(p)}$ should be ranked higher than $q_j^{(p)}$, denoted as the partial order $q_i^{(p)} \succ q_j^{(p)}$.

Suppose F is the set of ranking functions, then each instance of it $f \in F$ can rank the pair $q_i^{(p)} \succ q_j^{(p)} \Leftrightarrow f(x_i^{(p)}) > f(x_j^{(p)})$. The training process aims to find the optimal f that can fit as many pairs of partial orders in the training set as possible.

Any pairwise[3] learning to rank algorithms, such as Ranking SVM[12], RankBoost[9], and RankNet[4], can be adopted to learn the ranking function f, in the above setting. For example, when using Ranking SVM, we assume f to be a linear combination of features $f(x) = \omega^T x$ (where ω is the parameter vector representing the weights of the features), and use the following optimization problem to learn the parameter ω,

$$\min_{\omega,\xi} \frac{1}{2}\|\omega\|^2 + C\sum_p \sum_{x_i^{(p)} \succ x_j^{(p)}} \xi_{ij}^{(p)}$$

$$s.t. \quad \omega^T(x_i^{(p)} - x_j^{(p)}) > 1 - \xi_{ij}^{(p)}, \forall x_i^{(p)} \succ x_j^{(p)}, \xi_{ij}^{(p)} \geq 0. \quad (1)$$

Here $\|\cdot\|$ is the L_2 norm, ξ is a set of slack variables, and C is a trade-off coefficient. If the solution to (1) is $\bar{\omega}$, then the ranking function can be written as

$$f(x) = \bar{\omega}^T x. \quad (2)$$

This ranking function will be used to rank the candidate queries for a new page in the user browsing behavior data. Then the top-ranked queries can be presented to users as *SearchTrigger* suggestions.

3.4 Diversification of SearchTrigger Queries

As shown in Section 2.3, *SearchTrigger* queries can be very diverse even if they are triggered by the same page. In order to minimize the dissatisfaction of a random user after seeing the suggested *SearchTrigger* queries, one needs to diversify the queries before presenting them to users[4].

To this end, our task is to select a subset S'_p ($S'_p \subseteq S_p$), which contains *SearchTrigger* queries that are diverse in their topics. We formulate this task as a *set selection* problem inspired by [10].[5] In particular, we define an objective $g(\cdot)$, as a function of ranking model $f(\cdot)$ (e.g., learned in the previous subsection) and a query dissimilarity measure $\delta(\cdot,\cdot)$. The goal is to select a set of queries, $S'_p \subseteq S_p$, such that the objective function $g(\cdot)$ can be maximized, i.e.,

$$S'_p = \arg \max_{S'_p \subseteq S_p, |S'_p| = m'} g(S'_p, f(\cdot), \delta(\cdot,\cdot)). \quad (3)$$

where $m' = |S'_p|$ is the size of S'_p.

A simple yet reasonable objective function is given as follows ($\lambda > 0$ is a trade-off coefficient),

$$g(S'_p, f(\cdot), \delta(\cdot,\cdot)) = \sum_{q_i^{(p)}, q_j^{(p)} \in S'_p} \delta(i,j) + \lambda \sum_{q_i^{(p)} \in S'_p} f(i). \quad (4)$$

It is clear that the maximization of this objective function will guarantee that the queries selected will have a large ranking score (since the sum of the ranking scores has been maximized), and each two queries will be different (since the average pairwise dissimilarity has been maximized).

[3]Besides pairwise learning to rank algorithms, one can also choose pointwise [8] and listwise [5] learning to rank algorithms.

[4]This can be regarded as a dual problem of search result diversification.

[5]Note that one can use other diversification formulations like those discussed in [7, 15].

To solve the above optimization problem, one needs to address two technical challenges. First, since queries are usually very short, it is non-trivial to define an effective query dissimilarity measure. Second, the problem is a typical NP-hard problem and thus the efficient optimization of it is non-trivial. We will present our solutions to these two challenges in the following subsections.

3.4.1 Query Dissimilarity Measure

To compute effective query dissimilarity measure, we propose using the page-query bipartite graph built in Section 3.2, since it contains rich information of query relationships. However, this graph is not fully reliable since many edges in the bipartite graph do not correspond to *SearchTrigger* patterns. This might not be a big issue for ranking model learning since it is a supervised process and we can leverage other features to avoid the negative influence of this graph. However, it may become a problem when we use the graph for query diversification, since this is an unsupervised optimization process. If the graph is unreliable, the optimization results will accordingly become unreliable.

To tackle the problem, we clean the graph before using it to compute query dissimilarity. For each page in the graph, we extract features for its co-occurrence queries and compute the ranking scores of these queries using the model learned in Section 3.3. After that, we normalize the scores to interval $[0, 1]$ and use the normalized scores to re-weight the corresponding edges in the bipartite graph. That is, for edge e_{ij} with original weight w_{ij}, if the normalized ranking score of query q_j with regards to page p_i is v_{ij}, we will change the edge weight of e_{ij} to $w_{ij}v_{ij}$. In this way, the bipartite graph is cleaned because the weights of the *SearchTrigger* patterns are enlarged and those non-SearchTrigger patterns are reduced.

We then calculate query dissimilarity by Jensen-Shannon divergence (JSD) [23] on the cleaned graph.[6] The basic idea is that if two queries share many pages with high weights, they should be similar to each other; otherwise dissimilar. According to the bipartite graph, we can represent each query $q_i^{(p)}$ as a vector β_i. Each dimension of the vector $\beta_{ij} = w_{ij}v_{ij}$, where j is the index of a page in the graph. Given two queries $q_i^{(p)}$ and $q_k^{(p)}$, their dissimilarity in terms of JSD is calculated as below,

$$\delta(i,k) = (D(\beta_i \| \alpha) + D(\beta_k \| \alpha))/2, \quad (5)$$

where $D(\cdot\|\cdot)$ is the Kullback-Leibler divergence [22] and $\alpha = (\beta_i + \beta_k)/2$.

3.4.2 Efficient Optimization

Let η be an indicator vector defined as below,

$$\eta_i = \begin{cases} 1, & q_i^{(p)} \in S'_p \\ 0, & q_i^{(p)} \notin S'_p \end{cases} \quad (i = 1, 2, \cdots, m). \quad (6)$$

Then the objective function $g(\cdot)$ can be written as,

$$g(S'_p, f(\cdot), \delta(\cdot,\cdot)) = \sum_{q_i^{(p)}, q_j^{(p)} \in S_p} \delta(i,j)\eta_i\eta_j + \lambda \sum_{q_i^{(p)} \in S_p} f(i)\eta_i. \quad (7)$$

[6]Note that one can use other forms of query dissimilarity, such as those based on cosine similarity and Pearson correlation [21]. Here the use of JSD is just an example.

Suppose $\Delta = \{\delta(i,j)\}$ represents the query dissimilarity matrix, then we can get the following equivalent form of the original optimization problem,

$$\begin{aligned} &\max \frac{1}{2}\eta^T \Delta \eta + \lambda f \eta \\ &s.t.\ e^T \eta = m';\ \eta = \{\eta_i\},\ \eta_i = 0, 1; \\ &e = (1, 1, \cdots, 1)^T;\ f = \{f_i\}. \end{aligned} \tag{8}$$

The above optimization problem is a typical 0-1 integer programming problem, which is NP-hard. We propose relaxing the values of η to be continuous (i.e., $\eta_i \in [0, 1]$), and converting (8) to the following quadratic optimization problem. Note that the same trick has been widely used in semi-supervised learning and spectral clustering [6][16].

$$\begin{aligned} &\min \frac{1}{2}\eta^T L \eta - \lambda f \eta \\ &s.t.\ e^T \eta = m';\ \eta = \{\eta_i\},\ 0 \leqslant \eta_i \leqslant 1; \\ &e = (1, 1, \cdots, 1)^T;\ f = \{f_i\}. \end{aligned} \tag{9}$$

Here $L = I - D^{-\frac{1}{2}} \Delta D^{-\frac{1}{2}}$, in which I is the identity matrix and D is a diagonal matrix with its diagonal elements equal to the sum of all the elements in the corresponding rows of Δ. Suppose the solution to the above optimization problem is η^*. Then we can select the queries corresponding to the largest m' elements in η^* to form the suggested query set. Actually, the above optimization problem has the following properties.

- It is not difficult to verify that L is a positive semi-definite matrix and thus (9) is a convex optimization problem. As a result, η^* is the global optimal solution to (9). In contrast, in some previous work like [10], a greedy method was used to solve similar set selection problem, which is not guaranteed to result in an optimal solution.

- The problem can be solved in a time complexity of $O(m^3)$. For each page, the number of candidate queries is usually not very large (e.g., less than 100). Therefore, the computational complexity turns out to be affordable.

Note that when users go to a search engine with the suggested *SearchTrigger* queries, the page that they previously browsed can serve as an informative context for the search engine. There is a rich literature of contextual information retrieval, which basically leverages various contextual information to improve search quality [18]. Many ideas in the previous work can be used directly or indirectly. In Section 4.3, we tested a simple contextual retrieval algorithm and the experimental results clearly demonstrated the benefit of using the aforementioned contextual information to answer *SearchTrigger* queries.

4. EXPERIMENTAL EVALUATION

In this section, we presented our experimental study on the proposed approach.

4.1 Datasets

We used the user browsing behavior data as mentioned in Section 2 for our experiments. After partitioning the data to sessions, we extracted 167,570,019 "browse → search" patterns from them. The distribution of page numbers and the numbers of "browse → search" patterns in the sessions are shown in Figure 2 and Figure 3. We filtered out the patterns whose queries contain non-alphanumeric terms like Chinese or Arabic words. Then we removed those pages whose number of occurrences in the data is less than 5, for most of them correspond to rare patterns. After the cleaning, we obtained 56,929,950 patterns left and built a page-query bipartite graph from these patterns. The graph contains 3,529,910 unique pages and 25,677,960 unique queries. The degree distribution of queries in the graph is shown in Figure 4. This bipartite graph was used to extract query features and compute query dissimilarities.

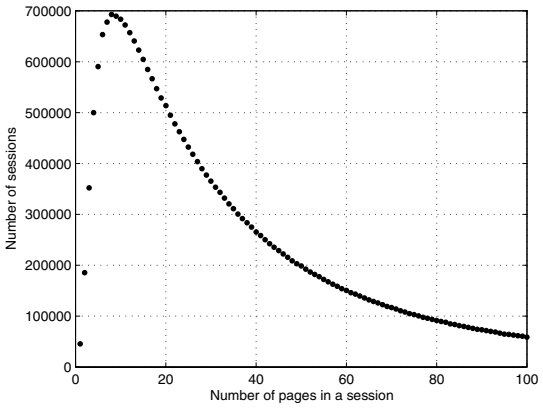

Figure 2: The distribution of page numbers in the sessions.

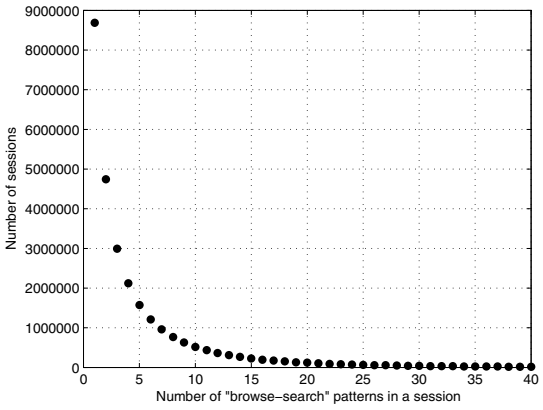

Figure 3: The distribution of the numbers of "browse → search" patterns in the sessions.

4.2 Performance of SearchTrigger Query Suggestion

We compared our proposed approach with some baseline methods, and investigated the benefit of diversifying suggested queries. All the methods under comparison are listed as below.

Key Phrase Extraction (KPE). KPE is a technique to extract important keywords or phrases from a given text document [11, 14, 19]. We use the method described in [19] to

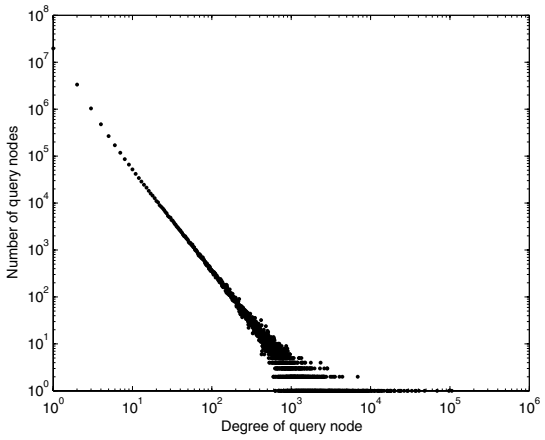

Figure 4: The degree distribution of the queries in the page-query bipartite graph.

extract key phrases in a page and regard them as triggered queries.

Pattern Frequency (PF). After extracting all "browse → search" patterns from user browsing behavior data, a straightforward solution is to count the frequencies of all queries that co-occur with a page, and suggest the most frequently-asked queries to users.

Query Ranking based on Query Features (QRQF). This method uses the features introduced in Section 3.2, employs Ranking SVM to combine them for query ranking. No diversification is introduced, and the ranking result given by Ranking SVM is directly suggested to users. The trade-off coefficient C is empirically set to 5.

SearchTrigger Queries Diversification (SQD). Based on the ranking result given by QRQF, the diversification method described in Section 3.4 is used to obtain a refined query set. The trade-off coefficients C and λ are empirically set to 5 and 2. This algorithm is exactly our proposed approach.

As mentioned in Section 2, we have two labeled datasets. The first dataset contains the labels of 849 "browse → search" patterns, and the second one contains the labels of all the queries with regards to 100 pages. We used the first dataset to train the models of QRQF and SQD, and then used the second one to test the performance of all the algorithms. Table 4 shows some examples of the suggested query sets produced by different algorithms. Pages No. 1, 2, and 3 correspond to `http://movies.about.com/od/currentfilms/`, `http://nationalpriorities.org/index.php?option=com_wrapper&Itemid=182`, and `http://pds.jpl.nasa.gov/planets/welcome.htm`, respectively. Due to space restrictions, for each page, only the top-5 ranked queries are shown. From the examples, we have the following observations:

1. The key phrases extracted from page content are very different from the queries issued by users. Some key phrases like *planetary exploration* can be well understood by viewing the page content and thus users may not want to learn more about them; some key phrases like *Hollywood* are well-known words and users seldom issue them as search queries; some extracted key phrases like *links within this site* seem to be neither a good summary of the page nor a possible triggered query.

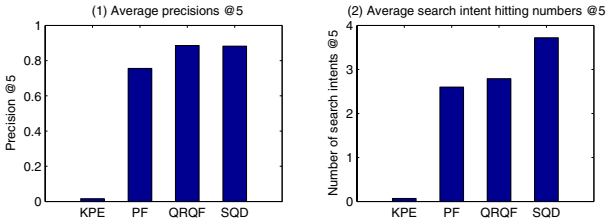

Figure 5: Average precisions and average search intent hitting numbers for different algorithms.

2. PF sometimes produces unexpected queries. For example, *rick maze* was suggested for page No.2. This query is the name of an editor in the magazine of Army Times. We searched within the whole site of page No.2 and did not find any convincing evidence to support that this query was triggered by the page. By further investigation on the browsing behavior data, we found the query corresponds to typical repeated search in several sessions.

3. The queries suggested by QRQF look more reasonable than those suggested by KPE and PF. However, many of the queries suggested by QRQF are very similar to each other, e.g., *recent movies*, *recent movie releases*, and *new movies releases*. By using SQD, we obtained even better results which are both reasonable and diverse. For example, The results produced by SQD for the planet page No.3 is a good example to demonstrate this.

To make a statistical comparison among these algorithms, for each page, we computed the precision [2] of the query set produced by each algorithm according to the ground-truth set labeled by human annotators; we also counted the number of search intents that the query set hit. After that, we computed the average precision and average hitting number for each algorithm. The results are shown in Figure 5, where the two measures are computed with respect to the suggested sets of the top 5 queries. We can see that QRQF and SQD correspond to the largest average precisions and average search intent hitting numbers, which are significantly better than the other two algorithms. Compared with QRQF, SQD performed significantly better in hitting more search intents, with only a small loss of precision. Therefore, we say that SQD is able to satisfy more users' information needs.

To better understand the benefit of diversifying the suggested query set, let us have a look at Figure 6. Instead of presenting the average results as in Figure 5, in Figure 6 we plot the distribution of pages with regards to different precisions and hitting numbers. In particular, each page p_i is represented by a two-dimensional vector (ϕ_i, φ_i), where ϕ_i is the precision and φ_i is the hitting number of the top-5 queries produced by an algorithm. In each sub-figure, X-axis corresponds to ϕ_i, Y-axis corresponds to φ_i, and Z-axis corresponds to the frequency of (ϕ_i, φ_i) (denoted as $\rho_{(\phi_i,\varphi_i)}$). The curved surface is fitted upon the tuples $(\phi_i, \varphi_i, \rho_{(\phi_i,\varphi_i)})$. From the sub-figures, we can clearly see the advantage of SQD over QRQF: its peak lies in the area with larger hitting number than that of QRQF, while their precisions are similar to each other.

Table 4: Examples of suggested query sets by different algorithms (top 5 results).

No.	1	2	3
KPE	Movies Movie News Hollywood Movies Hollywood New on Video	NPP Cost of War Obama Afghanistan Billion	planetary planetary exploration planetary exploration program Welcome to the Planets links within this site
PF	recent movies new movies releases new movies new movies released sports movies	iraq war rick maze positives of iraq war iraq war cost war in iraq deaths	planets pictures of the planets pictures of space shuttle photos of planets saturn photos
QRQF	recent movies movies recent movie releases dancing movies new movies releases	iraq war iraq war cost information on the war in iraq cost of iraq war war in iraq cost	planets 9 planets photos of planets planets solar system pictures of the planets
SQD	new movies releases recent movies dancing movies sports movies movies of 2006	cost of iraq war chart iraq war coalition of the willing members war in iraq deaths current iraq war debt	planets nasa kids pictures saturn photos pictures of space shuttle ufos

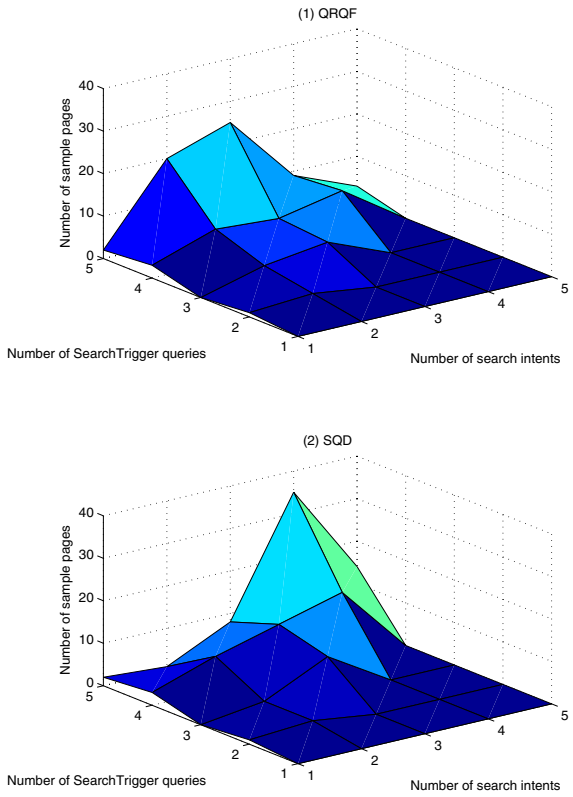

Figure 6: The performance comparison of QRQF and SQD.

4.3 Contextual Re-ranking

To verify whether it is beneficial to use the triggering pages as contextual information for search, we tested a simple contextual retrieval algorithm in this subsection.

The basic idea is to extract some additional features to represent the content similarity between the triggering page and the documents to rank, and use these features to rerank the original search results. Specifically, suppose a user selects a suggested *SearchTrigger* query q for page p, then both q and p will be sent to the search engine.[7] Given query q, the search engine can retrieve top-k relevant documents, i.e., $D = \{t_1, t_2, \cdots, t_k\}$, using its default retrieval function. After that, the content similarities between page p and these documents are calculated. According to the similarities, the similar documents to p will be promoted in the search result, as compared to those equally relevant but dissimilar documents. This heuristic is designed by considering that query q is triggered by page p and thus the user might be willing to see documents sharing similar content or topic with the triggering page p. The re-ranking algorithm is described in Table 5, where $sim(\cdot)$ is the cosine similarity function and γ is a parameter to set the weight of contextual information (in our experiments, we empirically set $\gamma = 0.4$).

To test the above algorithm, we designed the following experiment. We used the model learned by SQD to test unlabeled pages in the cleaned bipartite graph. If a "browse → search" pattern is predicted as *SearchTrigger* and the user did click a URL in the search result given by a search engine SE for the query (which can be observed in user browsing behavior data), we will regard it as a "browse → search → click" pattern. We sampled 500 such patterns from the "browse → search" sessions. For each of these patterns, we submitted that query to search engine SE and got the top-50 pages in its search result. We crawled the content

[7]Actually only the URL of the page needs to be sent to the search engine. With the URL, one can quickly retrieve the content of the page from the index of the search engine.

Table 5: Contextual re-ranking for search.

Input: triggering page p, SearchTrigger query q, and weighting parameter γ; **Output**: re-ranked documents $\{t'_1, t'_2, \cdots, t'_k\}$;
1. Retrieve the documents $D = \{t_1, t_2, \cdots, t_k\}$ of q using search engine's default retrieval function. The documents are sorted in the descending order of their relevance scores $\{r_1, r_2, \cdots, r_k\}$; 2. For each document $t_i \in D$ do $s_i = sim(p, t_i)$; $s'_i = r_i + \gamma s_i$; 3. Sort documents in D to $\{t'_1, t'_2, \cdots, t'_k\}$ in the descending order of s'_i;

of these pages and used the algorithm in Table 5 to rerank these pages. We regard the clicked page in the session as ground truth. If the reranking algorithm really boosted this clicked page, we say there is gain for this pattern. Among the 500 patterns, our experimental results show that there are 337 patterns (or 67.4%) with gains, 22 patterns (or 4.4%) without any position change, and 141 patterns (or 28.2%) with losses. This demonstrates that the use of contextual information can improve search quality and user satisfaction.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed the concept of actively predicting users' search intents based on their browsing behaviors. Our analysis on large scale user browsing behavior data indicates that many search intents are triggered by the pages that a user browses right before his/her search actions. In order to suggest meaningful queries to satisfy such intents of users when they browse the Web, we have proposed a machine learning method and demonstrated its effectiveness in the scenario of contextual retrieval. Our experimental results have shown that the proposed approach can predict meaningful queries to users for a given page, and can significantly improve the search quality with regards to these queries.

For future work, we would like to study the case that a query is triggered by a sequence of successively browsed pages, and the case that a page triggers several queries with different intents of a user in the same session. We believe that the deep understanding of users' search intents when they are browsing can help extend the functionality of search engines beyond their current boundaries, and can also provide users with a much better experience on the Web.

6. REFERENCES

[1] R. A. Baeza-Yates, C. A. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *EDBT Workshops*, pages 588–596, 2004.

[2] R. A. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. Addison Wesley, 1999.

[3] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *KDD*, pages 875–883, 2008.

[4] C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender. Learning to Rank using Gradient Descent. In *ICML*, pages 89–96, 2005.

[5] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In *ICML*, pages 129–136, 2007.

[6] O. Chapelle, B. Scholkopf, and A. Zien. Semi-Supervised Learning. MIT Press, Cambridge, MA, 2006.

[7] C. L.A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Buttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR*, pages 659–666, 2008.

[8] D. Cossock and T. Zhang. Subset ranking using regression. In *COLT*, pages 605–619, 2006.

[9] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research, 2003 (4).

[10] S. Gollapudi and A. Sharma. An Axiomatic Approach for Result Diversification. In *WWW*, pages 381–390, 2009.

[11] M. P. Grineva, M. N. Grinev, and D. Lizorkin. Extracting key terms from noisy and multitheme documents. In *WWW*, pages 661–670, 2009.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

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

[14] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *CIKM*, pages 233–242, 2007.

[15] F. Radlinski, R. Kleinberg, and T. Joachims. Learning Diverse Rankings with Multi-Armed Bandits. In *ICML*, 2008.

[16] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888-905, 2000.

[17] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *SIGIR*, pages 159–166, 2007.

[18] R. W. White, P. Bailey, and L. Chen. Predicting user interests from contextual information. In *SIGIR*, pages 363–370, 2009.

[19] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *ACM DL*, pages 254–255, 1999.

[20] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to Ad Hoc information retrieval. In *SIGIR*, pages 334-342, 2001.

[21] http://en.wikipedia.org/wiki/Correlation

[22] http://en.wikipedia.org/wiki/Kullback-Leibler_divergence

[23] http://en.wikipedia.org/wiki/Jensen-Shannon_divergence

[24] http://searchengineland.com/nielsen-netratings-august-2007-search-share-puts-google-on-top-microsoft-holding-gains-12243

[25] http://www.accuracast.com/search-daily-news/seo-7471/us-search-engine-market-share-data-jan-2009/

[26] http://www.comscore.com/Press_Events/Comunicados_de_prensa/2007/node_1285/Top_US_Search_Engines