Title: Web Log Analysis

Name: Matthias Hagen, Benno Stein

Affil./Addr.: Bauhaus-Universität Weimar

99421 Weimar, Germany

<first name>.<last name>@uni-weimar.de

Web Log Analysis

Synonyms

query log analysis, user interaction, search engine, query sessions, search missions, query suggestion, query auto-completion, query spelling correction, query expansion, query segmentation, learning to rank, caching, partitioned index, sharding

Glossary

web log: File containing the time-stamped interactions of users with a search engine (i.e., queries and clicks).

query: Keywords submitted to a search engine.

click: Selecting a search result for further inspection.

index: A data structure to serve fast lookups. For web search typically implemented as an inverted index that matches keywords to documents that contain the keyword.

posting list: One line of an inverted index (i.e., for a given keyword, the documents that contain it).

Definition

Web log analysis is the task of mining interesting patterns from the user interactions logged by a web search engine. Such log files usually contain time-stamped submitted

queries and clicks on results. At search engine site, analyzing these interactions gives insights into user behavior and also helps to improve effectiveness and efficiency of the service.

Introduction

Web search engines nowadays handle billions of queries per day. Logged queries and subsequent user interactions at this scale is an invaluable source of information. It allows for analyzing the "average" characteristics of usage like query length or click behavior. Even more important, query log analysis helps the search engine to improve its effectiveness (e.g., query suggestions or spelling corrections mined from a log) as well as its efficiency (e.g., result caching for frequent queries). Thus, log analysis and mining interesting patterns is heavily applied at search engine site. We review the main directions and also briefly introduce or reference the underlying algorithmic concepts. For a more thorough overview, we refer to the monograph by Silvestri, which covers many of the presented topics in greater detail and with more references (Silvestri, 2010).

Key Points

Query log analysis has three main goals. First, it is used to gain insights into the characteristics of submitted queries and subsequent interactions such as query reformulation patterns. Second, it allows to improve search effectiveness such as query suggestion or learning to rank; and third, system efficiency can be influenced by deriving caching strategies or by index sharding.

Typically, web queries are rather short (2–3 keywords) and follow a long tail distribution, i.e., few queries are very frequent, many queries submitted only once. The clicking behavior naturally prefers the first results, and only very few users view

more than 10 results at all. From similar queries in the log, search engines can identify potential expansion terms for query suggestions and even spelling correction hints. The clicked links show which results are preferred over others by the users, rising the field of learning to rank. With respect to efficiency, results of frequently submitted queries can be cached to faster serve the user and improve the overall search experience.

Historical Background

The first available web query logs were published in the late 1990's. In 1997, the once famous search engine Excite made available a sample from its query log. Later followed AltaVista and TodoBR. However, in 2006 a scandal surrounded the public release of a three-month query log by AOL (Pass et al, 2006): From the released queries it was possible to identify specific users, which started a lasting debate on privacy concerns related to query logs. As a consequence, none of the larger commercial search engines has publicly released larger portions of non-anonymized queries since then. Typically, publicly available logs now only contain queries and clicks anonymized as IDs without query strings or result URLs. One example is the Yandex log used for the challenge at the WSDM 2012 workshop on Web Search Click Data (`http://imat-relpred.yandex.ru/en/`, last accessed: August 29, 2012). Nevertheless, even without recent publicly available non-anonymized query logs, analyzing user querying and click behavior is of utmost importance for search engines and has a high impact on running the services.

Characteristics of Search Engine Interaction

We briefly review important findings on querying and click behavior as well as longer user interaction sessions involving more than just one query.

Queries

The typical web search query is rather short (average between 2–3 keywords) which is consistent over many studies and query logs. In recent years, queries tend to get somewhat longer, but no dramatic effect yet. Still about 40–50% of the queries contain just one or two keywords, and overall the query length distribution follows a power law (Arampatzis and Kamps, 2008).

Markatos also observed a power law distribution when analyzing query frequency (how often is the same query submitted) (Markatos, 2001). For instance, about 16–20% of the queries Google receives have not been submitted before (Mitchell, 2012). On the other end, there are a bunch of very popular queries that are submitted again and again such as celebrity names or current trends. With respect to query topics, several studies show that queries for entertainment, shopping, or porn are the most frequent with 10–15% each (Beitzel et al, 2007). The goal of a query is typically divided into three categories: about 20% are navigational (reaching a specific page), about 50% are informational (finding information on a topic), and about 30% are transactional (perform a transaction like buying something) (Broder, 2002). When analyzed by submission time, it turns out that most queries are submitted in the late afternoon or evening. For instance, compared to the queries submitted around 9am, the time slot around 5pm achieves twice as many queries (Pass et al, 2006).

Clicks

Besides the queries itself, also clicks on results are part of users' interactions with a search engine. Not that surprisingly, the top positions in the ranking attract much more clicks than the ones further below. Almost for half of the queries the first result is clicked, while only about 15% click on the second rank (Joachims et al, 2007). Only very few queries receive a click on a result below the top 10 ranks (i.e., users rarely go

to the second page of results), but when users view the second page the probability to also view the third page is quite high.

By using the clicked documents of queries, Baeza-Yates and Tiberi form a social network of queries. Thereby, two queries are connected if they lead to similar clicks (Baeza-Yates and Tiberi, 2007). Such networks can for instance be interesting to identify synonyms among the keywords in queries that often lead to clicks on the same documents. A similar idea is to build a query folksonomy from the click-through graph (Benz et al, 2010; Francisco et al, 2012).

Sessions

Many queries are answered so well by web search engines, that one of the top results already satisfies the user's underlying information need. There are however also a non-negligible number of cases where the first query does not yield appropriate results. The user then typically reformulates the query and submits a new one. Such an interaction pattern forms a query session—queries submitted for the same information need. Identifying such sessions is important for search engines. By using the session knowledge they could offer help to a user stuck with her information need, or the engine could simply try to improve results for the last query in a session using the previous ones (which is the task in the TREC Session Track). As for session detection, the first proposed methods were simply time based (i.e., whenever the time passed between two interactions exceeds a given threshold, a new session is started) but more modern methods also take different features of query similarity into account (Hagen et al, 2011b). Again, like for query length or popularity, it turns out that most sessions are rather short (about 40% contain just one query) with an average session length between 2–3 queries.

Identifying which queries often appear together in sessions and linking them by edges yields the query-flow graph (Boldi et al, 2008) that contains paths for typical querying chains that users follow.

Effectiveness

Knowing what queries were submitted in the past offers a great opportunity to improve a search engine's effectiveness. Hereby, we mean improving a user's search experience by providing better tailored results. For instance, very similar queries might be a source for spelling correction and the users' clicks can improve the ranking function in a learning to rank manner.

Query Spelling Correction

Having a typo in a query might result in an empty result page. As such would probably distract a user, search engines apply spelling correction to queries whenever it seems appropriate. However, note that due to the restricted length of queries, there is often not that much context to base a correction on. Instead, using previously submitted queries is the key. But the straightforward idea of simply building a dictionary from the keywords in the most frequent queries in a log and applying the traditional dictionary based spelling correction (replace unknown words with their closest word in the dictionary) does not really work. The main problem is that for some prominent queries even misspellings are much more frequent than correct spellings for less frequent queries. Thus, the dictionary would often not include correct versions of rare queries.

A better way of exploiting query logs is to also include the context words in the dictionary. Such methods work on query level corrections that try to reformulate a potentially misspelled query in several steps into a correct one by always choosing the most likely correction (Cucerzan and Brill, 2004). To find such correction paths,

Cucerzan and Brill apply a Viterbi search over the potential corrections using two dictionaries. One contains trusted words as in traditional spelling correction, the other is built from the query log including frequency information to determine the likelihood of a correction. Note that their algorithm allows a word to be split up into two, but at every step of the correction path only one word is corrected (and not two or more simultaneously). More recent approaches even work in an online manner correcting a query while the searcher is typing (Duan and Hsu, 2011).

Query Autocompletion

To prevent misspelled queries right from the start, search engines can suggest typical completions of a query while a searcher types. This feature is known as autocompletion. It is not only useful to reduce misspellings but also for suggesting a typical query related to the string the user already typed. Approaches for autocompletion range from identifying similar queries from the user's previous queries in the query log (Bar-Yossef and Kraus, 2011) (using cosine similarity between queries) to completing the last keyword in a query with what is most likely from other users' queries (Bast and Weber, 2006). An interesting question then is that of how many potential completions to present to the user. Current search engines use different schemes and show from four up to eight possibilities.

Query Suggestion

Slightly different than autocompletion that is done at the time of typing a query, query suggestions are usually shown on the results page. The idea is to show queries related to the one the user submitted and thus to provide recommendations of what else could be explored. A naïve way would be to simply show queries from the log that share at least one term with the current query. As this obviously is a very broad notion

of query relatedness, it can often lead to unsolicited suggestions. A better and more sophisticated idea is to use previous sessions of other users. The suggestions then are queries that were submitted following a specific query in similar sessions and that resulted in a click (Baraglia et al, 2009). Assume for instance, that a query q in the query log is often followed by a query q', and that users often clicked results for q'. This then forms a strong evidence that whenever another user submitted q, presenting q' as a query suggestion is a good idea. Another approach combines this general knowledge of queries and click-through from a query log with the user's context in form of her previous queries (Cao et al, 2008).

Query Expansion

As web queries mostly are rather short with an average length between 2–3 keywords, it might often be beneficial to add a few more terms to make the query more specific and thus to retrieve more precise results. Another problem is the potential mismatch of a query's keywords with the vocabulary in the searched collection. Both issues are tackled by the task of query expansion, which seeks to automatically add promising keywords to an original query before the actual retrieval. When first developed, query expansion was not based on query logs, but the availability of large logs now adds another potential boost to be exploited. Cui et al build a graph G from the click-through information that reflects what documents were clicked for which query (Cui et al, 2002). The nodes of G are keywords, and for each keyword w in a document clicked for a query q and for each keyword w' in q, the graph contains an edge weighted by the probability of the click-through from queries containing w' to documents containing w. These keyword correlations in G are used to identify potential expansions among the keywords with high correlations to the keywords in a given query. Note that this is a form of pseudo-relevance feedback, which assumes that clicked documents of previous queries were

relevant for these queries. A related approach uses a reverted index, which stores for a document those queries that retrieve the document high in the ranking (Pickens et al, 2010). Potential expansion keywords for a new query q are the keywords associated in the reverted index which the documents q retrieves. One way of building the reverted index is to use a past query log.

Note however, that query expansion is rarely used at web scale due to performance issues of the expansion methods and also because query suggestion provides a very powerful and less "invasive" alternative.

Query Segmentation

For longer web queries it is often beneficial to know which keywords form compound concepts or phrases that should be retrieved as such. Most search engines allow to specify which parts of a query are indivisible by enclosing them in double quotes, but less than 1.12% of the submitted queries contain quotes or other operators (White and Morris, 2007). As the majority of users do not use quotes, an automatic solution often is applied to identify concepts and phrases in queries—the task of query segmentation.

Two recent approaches are based on query log analyses. One uses co-occurrence frequencies of keyword groups in logged queries (Mishra et al, 2011), the other exploits click-through information (Li et al, 2011) to identify potential phrases. However, the query-log-based approaches do not perform better than other state-of-the-art methods relying solely on web frequencies (Hagen et al, 2011a, 2012).

Learning to Rank

The clicks of past search engine users can be seen as a relevance judgment for a web page with respect to the query it receives clicks for. Search engines can exploit the clicks in their query logs to fine-tune the ranking function. In particular, a modern search

engine uses many features for its ranking function. Carefully balancing the weights of different features is usually done via machine learning techniques. To get training data for learning the ranking function, user clicks on documents are an invaluable resource. Joachims et al propose several different strategies of applying click-through information to learning to rank (Joachims et al, 2007). One of the basic ideas is to assume that clicked documents are more relevant than non-clicked documents ranked above them. Assume for instance that a user clicked on the first and third entry of a search result page. Then the third result can be seen as more relevant to the query than the second one. More sophisticated ideas also consider clicks for follow-up queries and assume that they represent more relevant results than non-clicked (or even clicked) documents for previous queries. Other interpretations suggest to consider only clicks for the top results as relevance indication.

The common theme of all click-relevance assumptions is that the extracted information should be used to train a model that optimizes the ranking. But note that it is always important to take into account the click bias of users (top results are clicked more often). Otherwise, top results receiving clicks potentially only benefit from their ranking position.

Efficiency

Besides search engine effectiveness that focuses on retrieval performance, efficiency considerations deal with the practical arrangements that influence runtime performance. As search engines typically run in a distributed environment, questions for index sharding (how to partition the document set) or caching (what results to keep in faster memory) naturally arise. We point to ideas of how query log information can be used in this respect.

Caching

Caching is a means to exploit the opportunities offered by typical memory hierarchies with several storage tiers (fast RAM, slower disks, etc.). The question always is: What portion of data to have on what tier in the hierarchy? As for query processing, a possible idea is to keep results of frequent queries in faster memory and thus serve a large portion of users without involving slower levels of the system. Instead of accessing the search engine's index that typically resides in slower memory due to its size, such cache hits (query results in the cache) can be directly served from the fast tier.

Important for caching is the fact that memory on faster levels is typically much smaller than that on slower levels. Hence, strategies are required that decide what to remove from a cache, once it is full. Basic ideas range from the standard least frequently used (LRU, removing the cache entry that was not accessed the longest) to combinations of caches were one keeps accessed items in an LRU manner, and another protected segment is reserved for entries that have resulted in a cache hit before (SLRU). As for caching results of a query (e.g., the top 10 results) it has been shown that SLRU is slightly better than LRU especially for smaller cache sizes (Markatos, 2001). However, neither LRU nor SLRU exploit query log information. That such log information can yield superior performance is for instance demonstrated by static dynamic caching (SDC) (Fagni et al, 2006). SDC mixes a static cache of frequent queries found via query log analyses and a dynamic cache for recent queries. Furthermore, SDC supports prefetching (i.e., storing not just the top 10 results but also often requested further results). Using the LRU strategy for the dynamic cache part, SDC outperforms other approaches with respect to hit-ratio while also achieving very good runtime performance. Note that from time to time it might be necessary to re-populate the static portion using more recent log data as otherwise the underlying frequencies get dated.

Besides caching complete results for queries, another idea is to cache raw posting lists for keywords. The posting lists are typically used for computing query results (e.g., by merging the lists of different keywords). One problem is that then the cache entries can have very different sizes while for result caching a good policy might be to have batches of 10 results (the top 10, rank 11 to 20, etc.) as typically search engines only show 10 results at once. Posting lists however can get really big for popular terms. Obviously, there is a trade-off for storing long lists of very popular terms or storing shorter lists of less popular ones. Baeza-Yates and Saint-Jean propose a corresponding strategy that assigns weights to posting lists according to the ratio of the keyword's frequency in the query log over the size of the posting list (Baeza-Yates and Saint-Jean, 2003). A static version of this method has the advantage of cache population in a preprocessing step such that there are no additional cache management costs at runtime.

Index Sharding

The index of a search engine itself plays a crucial role at runtime. Whenever there is no cache hit for query results or a posting list, the index has to be accessed. But typically the size of the indexed collection precludes a search engine to have the complete index on one machine. Instead, modern search engines are distributed over many machines. This means that also the index has to be partitioned which is typically done by randomly assigning documents to be served by a specific machine. However, log information can help to partition in a better way. Assume for instance that it is known which queries retrieve which documents. A document could then be represented by a vector containing the queries that are related to the document (Puppin and Silvestri, 2006) (also note the similarity to the above mentioned reverted index). The proposed query vector based method co-clusters the documents and queries from a log into clus-

ters of documents retrieved for similar queries and clusters of queries retrieving similar documents. The document clusters form the partitioning of the collection while the query clusters can then be used to identify clusters that need to be accessed for answering a given query. An interesting side effect is that the clustering can also identify the documents that are not among the results for any query and thus form a separate cluster. This is in line with the idea of retrievability of documents (Azzopardi and Vinay, 2008). However, the main drawback of the outlined partitioning idea is the cost of clustering in case of large collections and that servers in the system that serve very prominent queries receive a very high load while others might even be idle.

Key Applications

As presented in this short survey, the main applications of query log analysis are to understand typical user behavior (query length, sessions), to improve search effectiveness (spelling, suggestions), and to improve search efficiency (caching, sharding).

Future Directions

Although research in the domain of query log analysis has yielded very powerful techniques to improve search experience for many users, there are still many interesting problems to work on. Consider for instance research on query sessions. It is still open what forms a good support for users during sessions. Although the participants in the recent TREC Session Tracks could improve performance for some sessions, other sessions suffered from decreased result quality. And even more interesting, log analyses have shown that many users combine several sessions into longer search missions that run for quite some time (Jones and Klinkner, 2008; Hagen et al, 2013) (e.g., think of planning a vacation which may consist of looking for a flight, a hotel, things to do,

etc.). Identifying such missions and offering appropriate support to the user seems a promising future direction.

Another interesting aspect could be to evaluate the size of a query log needed for specific tasks. While the big commercial search engines can easily collect huge query logs, smaller competitors, site search engines, or even academic researchers do not have access to such large scale logs. In fact, the non-availability of public large scale logs is one of the big problems for academic research in the query log analysis field (although the underlying privacy reasons are understandable). It could therefore be very interesting to examine what can be done with smaller logs in a bootstrapping manner for problems like learning to rank from clicks, query suggestions, or session support.

Cross-References

00382 Analysis and Mining of Tags, (Micro-)Blogs, and Virtual Communities

00138 Clustering Algorithms

00056 Data Mining

00141 Distance and Similarity Measures

00110 Folksonomies

00178 Machine Learning for Social Networks

00353 Microtext Processing

00377 Query Answering in the Semantic Social Web: An Argumentation-based Approach

00117 Retrieval Models

00188 Social Media Mining and Knowledge Discovery

00261 Social Web Search

00391 Stream Querying and Reasoning on Social Data

00170 Theory of Probability, Basics and Fundamental

00171 Theory of Statistics, Basics and Fundamental

00104 World Wide Web

References

Arampatzis A, Kamps J (2008) A study of query length. In Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008), Singapore, July 20–24, 2008, pp 811–812

Azzopardi L, Vinay V (2008) Retrievability: an evaluation measure for higher order information access tasks. In Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008), Napa Valley, California, USA, October 26–30, 2008, pp 561–570

Baeza-Yates RA, Saint-Jean F (2003) A three level search engine index based in query log distribution. In Proceedings of the 10th International Symposium on String Processing and Information Retrieval (SPIRE 2003), Manaus, Brazil, October 8–10, 2003, Lecture Notes in Computer Science, vol 2857, pp 56–65

Baeza-Yates RA, Tiberi A (2007) Extracting semantic relations from query logs. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2007), San Jose, California, USA, August 12–15, 2007, pp 76–85

Bar-Yossef Z, Kraus N (2011) Context-sensitive query auto-completion. In Proceedings of the 20th International Conference on World Wide Web (WWW 2011), Hyderabad, India, March 28–April 1, 2011, pp 107–116

Baraglia R, Cacheda F, Carneiro V, Fernández D, Formoso V, Perego R, Silvestri F (2009) Search shortcuts: a new approach to the recommendation of queries. In Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys 2009), New York, NY, USA, October 23–25, 2009, pp 77–84

Bast H, Weber I (2006) Type less, find more: fast auto-completion search with a succinct index. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006), Seattle, Washington, USA, August 6–11, 2006, pp 364–371

Beitzel SM, Jensen EC, Chowdhury A, Frieder O, Grossman DA (2007) Temporal analysis of a very large topically categorized web query log. Journal of the American Society for Information Science and Technology 58(2):166–178

Benz D, Hotho A, Jäschke R, Krause B, Stumme G (2010) Query Logs as Folksonomies. In Datenbank-Spektrum 10(1):15-24

Boldi P, Bonchi F, Castillo C, Donato D, Gionis A, Vigna S (2008) The query-flow graph: model and applications. In Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008), Napa Valley, California, USA, October 26-30, 2008, ACM, pp 609–618

Broder AZ (2002) A taxonomy of web search. SIGIR Forum 36(2):3–10

Cao H, Jiang D, Pei J, He Q, Liao Z, Chen E, Li H (2008) Context-aware query suggestion by mining click-through and session data. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008), Las Vegas, Nevada, USA, August 24–27, 2008, pp 875–883

Cucerzan S, Brill E (2004) Spelling correction as an iterative process that exploits the collective knowledge of web users. In Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), Barcelona, Spain, July 25–26, 2004, pp 293–300

Cui H, Wen JR, Nie JY, Ma WY (2002) Probabilistic query expansion using query logs. In Proceedings of the 11th International World Wide Web Conference (WWW 2002), Honolulu, Hawaii, USA, May 7–11, 2002, pp 325–332

Duan H, Hsu BJP (2011) Online spelling correction for query completion. In Proceedings of the 20th International World Wide Web Conference (WWW 2011), Hyderabad, India, March 28–April 1, 2011, pp 117–126

Fagni T, Perego R, Silvestri F, Orlando S (2006) Boosting the performance of web search engines: caching and prefetching query results by exploiting historical usage data. ACM Transactions on Information Systems 24(1):51–78

Francisco AP, Baeza-Yates RA, Oliveira AL (2012) Mining query log graphs towards a query folksonomy. In Concurrency and Computation: Practice and Experience 24(17):2179–2192

Hagen M, Potthast M, Stein B, Bräutigam C (2011a) Query segmentation revisited. In Proceedings of the 20th International World Wide Web Conference (WWW 2011), Hyderabad, India, March 28–April 1, 2011, pp 97–106

Hagen M, Stein B, Rüb T (2011b) Query session detection as a cascade. In Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM 2011), Glasgow, United Kingdom, October 24–28, 2011, pp 147–152

Hagen M, Potthast M, Beyer A, Stein B (2012) Towards optimum query segmentation: in doubt without. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM 2012), Maui, HI, USA, October 29–November 02, 2012, pp 1015–1024

Hagen M, Gomoll J, Beyer A, Stein B (2013) From search session detection to search mission detection. In Proceedings of the 10th International Conference Open Research Areas in Information Retrieval (OAIR 2013), Lisbon, Portugal, May 22–24, 2013, to appear

Joachims T, Granka LA, Pan B, Hembrooke H, Radlinski F, Gay G (2007) Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. ACM Transactions on Information Systems 25(2), article 7

Jones R, Klinkner KL (2008) Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008), Napa Valley, California, USA, October 26–30, 2008, pp 699–708

Li Y, Hsu BJP, Zhai C, Wang K (2011) Unsupervised query segmentation using clickthrough for information retrieval. In Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2011), Beijing, China, July 25–29, 2011, pp 285–294

Markatos EP (2001) On caching search engine query results. Computer Communications 24(2):137–143

Mishra N, Roy R, Ganguly N, Laxman S, Choudhury M (2011) Unsupervised query segmentation using only query logs. In Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28–April 1, 2011 (Companion Volume), pp 91–92

Mitchell J (2012) How Google search really works. ReadWriteWeb, URL `http://www.readwriteweb.com/archives/interview_changing_engines_mid-flight_qa_with_goog.php`, last accessed: May 6, 2013

Pass G, Chowdhury A, Torgeson C (2006) A picture of search. In Proceedings of the 1st International Conference on Scalable Information Systems (Infoscale 2006), Hong Kong, May 30–June 1, 2006, paper 1

Pickens J, Cooper M, Golovchinsky G (2010) Reverted indexing for feedback and expansion. In Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010), Toronto, Ontario, Canada, October 26–30, 2010, pp 1049–1058

Puppin D, Silvestri F (2006) The query-vector document model. In Proceedings of the 15th ACM Conference on Information and Knowledge Management (CIKM 2006), Arlington, Virginia, USA, November 6–11, 2006, pp 880–881

Silvestri F (2010) Mining query logs: Turning search usage data into knowledge. Foundations and Trends in Information Retrieval 4(1-2):1–174

White RW, Morris D (2007) Investigating the querying and browsing behavior of advanced search engine users. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), Amsterdam, The Netherlands, July 23–27, 2007, pp 255–262