Mining Query Logs:
Turning Search Usage Data into Knowledge
Mining Query Logs: Turning Search Usage Data into Knowledge

Fabrizio Silvestri

ISTI — CNR
via G. Moruzzi, 1
56124 Pisa
Italy
fabrizio.silvestri@isti.cnr.it

now
the essence of knowledge

Boston – Delft
Editorial Scope

Foundations and Trends® in Information Retrieval will publish survey and tutorial articles in the following topics:

- Applications of IR
- Architectures for IR
- Collaborative filtering and recommender systems
- Cross-lingual and multilingual IR
- Distributed IR and federated search
- Evaluation issues and test collections for IR
- Formal models and language models for IR
- IR on mobile platforms
- Indexing and retrieval of structured documents
- Information categorization and clustering
- Information extraction
- Information filtering and routing
- Metasearch, rank aggregation and data fusion
- Natural language processing for IR
- Performance issues for IR systems, including algorithms, data structures, optimization techniques, and scalability
- Question answering
- Summarization of single documents, multiple documents, and corpora
- Text mining
- Topic detection and tracking
- Usability, interactivity, and visualization issues in IR
- User modelling and user studies for IR
- Web search

Information for Librarians

ISSN paper version 1554-0669. ISSN online version 1554-0677. Also available as a combined paper and online subscription.
Mining Query Logs: Turning Search Usage Data into Knowledge

Fabrizio Silvestri

ISTI — CNR, via G. Moruzzi, 1, 56124 Pisa, Italy, fabrizio.silvestri@isti.cnr.it

Abstract

Web search engines have stored in their logs information about users since they started to operate. This information often serves many purposes. The primary focus of this survey is on introducing to the discipline of query mining by showing its foundations and by analyzing the basic algorithms and techniques that are used to extract useful knowledge from this (potentially) infinite source of information. We show how search applications may benefit from this kind of analysis by analyzing popular applications of query log mining and their influence on user experience. We conclude the paper by, briefly, presenting some of the most challenging current open problems in this field.
5 Enhancing Efficiency of Search Systems

5.1 Caching

5.2 Index Partitioning and Querying in Distributed Web search Systems

5.3 Summary

6 New Directions

6.1 Eye Tracking

6.2 Web Search Advertisement

6.3 Time-series Analysis of Queries

6.4 Summary

7 Conclusions

Acknowledgments

References
Introduction

“History teaches everything, even the future.”
— Alphonse de Lamartine, speech at Macon 1847.

Think about it, for a moment: after checking e-mails, and checking your favorite on-line newspaper and comic strip, what is the first thing you do when connected to the web? You probably open a search engine and start looking for some information you might need either for work or for leisure: news about your favorite actor, news about presidential candidates, and so on.

Even though they are quite rooted in our lives, web search engines are quite new on the scene.

Query Log Mining is a branch of the more general Web Analytics scientific discipline. Indeed, it can be considered a special type of web usage mining. According to the Web Analytics Association, “Web Analytics is the measurement, collection, analysis and reporting of Internet data for the purposes of understanding and optimizing Web usage.”

In particular, query log mining is concerned with all those techniques aimed at discovering interesting patterns from query logs of web search engine with the purpose of enhancing either effectiveness or efficiency of an online service provided through the web.
2 Introduction

Keeping into account that query log mining is not only concerned with the search service (from which queries usually come from) but also with more general services like, for instance, search-based advertisement, or web marketing in general [105].

1.1 Web Search Engines

Systems that can be considered similar to modern web search engines started to operate around 1994. The now-defunct World Wide Web Worm (WWW) [146] created by Oliver McBryan at the University of Colorado, and the AliWeb search engine [124] created by Martijn Koster in 1994, are the two most famous examples. Since then many examples of such systems have been around the web: AltaVista, Excite, Lycos, Yahoo!, Google, ASK, MSN (just to name a few). Nowadays, searching is considered one of the most useful application on the web.

As reported in 2005 by Pew Research Center for The People & The Press [161]:

“search engines have become an increasingly important part of the online experience of American internet users. The most recent findings from Pew Internet & American Life tracking surveys and consumer behavior trends from the comScore Media Metrix consumer panel show that about 60 million American adults are using search engines on a typical day” [188].

Even if this quote dates back to 2005, it is very likely that those survey results are still valid (if not still more positives for search engines). On the other side of the coin, search engines’ users are satisfied by their search experience [189].

In a paper overviewing the challenges in modern web search engines’ design, Baeza-Yates et al. [14] state:

The main challenge is hence to design large-scale distributed systems that satisfy the user expectations, in which queries use resources efficiently, thereby reducing the cost per query.
Therefore, the two key performance indicators in this kind of application, in order, are: (i) the quality of returned results (e.g. handle quality diversity and fight spam), and (ii) the speed with which results are returned.

Web search engines are part of a broader class of software systems, namely Information Retrieval (IR) Systems. Basically, IR systems were born in the early 1960s due to two major application needs. Firstly, allowing searching through digital libraries. Secondly, the need for computer users to search through the data they were collecting in their own digital repositories.

Intuitively, an IR system is a piece of software whose main purpose is to return a list of documents in response to a user query. Thus far, this description makes IR systems similar to what a DB system is. Indeed, the most important difference between DB and IR systems is that DB systems return objects that exactly match the user query, whereas IR systems have to cope with natural language that makes it simply impossible for an IR system to return perfect matches. Just to make a very simple example: what does meta refer to? A meta character? The meta key in computer keyboards? Every single query may mean different things to different users. Even worse, polysemy also happens. In Spanish the word meta means goal.

To this extent, a web search engine is in all respects an IR system only on a very large scale. The uncertainty in users’ intent is also present in web search engines. Differently from smaller scale IR systems, though, web IR systems can rely on the availability of a huge amount of usage information stored in query logs.

One of the most used ways of enhancing the users’ search experience, in fact, is the exploitation of the knowledge contained within past queries. A query log, typically, contains information about users, issued queries, clicked results, etc. From this information knowledge can be extracted to improve the quality (both in terms of effectiveness and efficiency) of their system. Figure 1.1 shows a fragment of the AOL query log. The format of this query log represents a record using five features: user id, query, timestamp, rank of the clicked result, host string of the clicked URL.
Introduction

Fig. 1.1 A fragment of the AOL query log [160].

How query logs interact with search engines has been studied in many papers. For a general overview, [12, 20] are good starting point references.

In this paper, we review some of the most recent techniques dealing with query logs and how they can be used to enhance web search engine operations. We are going to summarize the basic results concerning query logs: analyses, techniques used to extract knowledge, most remarkable results, most useful applications, and open issues and possibilities that remain to be studied.

The purpose is, thus, to present ideas and results in the most comprehensive way. We review fundamental, and state-of-the-art techniques. In each section, even if not directly specified, we review and analyze the algorithms used, not only their results. This paper is intended for an audience of people with basic knowledge of computer science. We also expect readers to have a basic knowledge of Information Retrieval. Everything not at a basic level is analyzed and detailed.

Before going on, it is important to make clear that all the analyses and results reported were not reproduced by the author. We only report
results as stated in the papers referenced. In some cases we slightly adapted them to make concepts clearer.

1.2 Sketching the Architecture of a Web Search Engine

A search engine is one of the most complicated pieces of software a company may develop. Consisting of tens of interdependent modules, it represents one of the toughest challenges in today’s computer engineering world.

Many papers and books sketch the architecture of web search engines. For example Barroso et al. [32] present the architecture of Google as it was in 2003. Other search engines are believed to have similar architectures. When a user enters a query, the user’s browser builds a URL (for example http://www.google.com/search?q=foundations+trends+IR). The browser, then, looks up on a DNS directory for mapping the URL main site address (i.e., www.google.com) into a particular IP address corresponding to a particular data-center hosting a replica of the entire search system. The mapping strategy is done accordingly to different objectives such as: availability, geographical proximity, load and capacity. The browser, then, sends an HTTP request to the selected data-center, and thereafter, the query processing is entirely local to that center. After the query is answered by the local data-center, the result is returned in the form of an HTML page, to the originating client.

Figure 1.2 shows they way the main modules of a web search engine are connected.

Web search engines get their data from different sources: the web (primarily), Image and video repositories (e.g. Flickr, or YouTube), etc. In particular, in the case of web content, a crawler scours through hypertext pages searching for new documents, and detecting stale, or updated content. Crawlers store the data into a repository of content (also known as web document cache), and structure (the graph representing how web pages are interconnected). The latter being used, mainly, as a feature for computing static document rank scores (e.g. PageRank [157], or HITS [122]). In modern web retrieval systems, crawlers continuously run and download pages from the web updating
incrementally the content of the document cache. For more information on crawling, interested readers can refer to Castillo’s Ph.D. thesis on web Crawling [57].

The textual (i.e., hypertextual) content is indexed to allow fast retrieval operations (i.e., query requests). The index (built by the Indexer) usually comprises of several different archives storing different facets of the index. The format of each archive is designed for enabling a fast retrieval of information needed to resolve queries. The format of the index is the subject of Section 5 where we review some of the most used techniques for optimizing index allocation policies.

Usually in real systems the design is tailored to favor aggregate request throughput not peak server response time [33].

In real-world search engines, the index is distributed among a set of query servers coordinated by a broker. The broker, accepts a query from the user and distributes it to the set of query servers. The index servers
1.2 Sketching the Architecture of a Web Search Engine

The typical structure of a distributed web search engine is shown in Figure 1.3. Relevant documents are retrieved, scores are computed, results are ranked, and then returned to the broker, which renders the result page and sends it to the user. Figure 1.3 illustrates the interactions among query servers and the broker.

The broker is usually where queries are grabbed and stored in the query logs. A module dedicated to analyze past queries is also usually available within the architecture components.

1.2.1 The Index

An Inverted File index on a collection of web pages consists of several interlinked components. The principal ones are the lexicon, i.e., the list of all the index terms appearing in the collection, and the corresponding set of inverted lists, where each list is associated with a distinct term of the lexicon. Each inverted list contains, in turn, a set of postings. Each posting collects information about the occurrences of the corresponding term in the collection’s documents. For the sake of simplicity, in the following discussion we consider that each posting
only includes the identifier of the document (DocID) where the term appears, even if postings actually store other information used for document ranking purposes (e.g. in the implementation [203] each posting also includes the positions and the frequency of the term within the document, and context information like the appearance of the term within specific html tags).

Several sequential algorithms have been proposed in the past, which try to balance the use of memory hierarchy in order to deal with the large amount of input/output data involved in query processing. The inverted file index [221] is the data structure typically adopted for indexing the web. This occurs for three reasons. First, an inverted file index allows the efficient resolution of queries on huge collections of web data [246]. In fact, it works very well for common web queries, where the conjunction of a few terms is to be searched for. Second, an inverted file index can be easily compressed to reduce the space occupancy in order to better exploit the memory hierarchy [203]. Third, an inverted file can be easily built using a sort-based algorithm in time complexity that is the same order of a sorting algorithm [246].

Query answering using inverted file is a very straightforward task. We illustrate the basic AND operation and refer to other papers for a thorough analysis of the remaining operations. Given a query as a conjunction of two terms ($t_1 \land t_2$), the query resolution proceeds by firstly looking up $t_1$ and $t_2$ in the lexicon to retrieve the corresponding inverted lists $l_1$ and $l_2$. The result set is then built by intersecting the two lists, thus, returning those documents having the two terms in common. During the intersection step a scoring function is also computed to evaluate the likeliness of a document to be relevant for the query. The top $r$ results are then selected (in typical web search engines $r$ is usually set to 10 results) and successively returned to the users who originated the query. Query processing can be done in two different ways: Document-At-A-Time (DAAT), when document lists for terms are scanned contemporary, as opposed to the Term-At-A-Time (TAAT) strategy, where each term is considered separately [219].

Another important feature of inverted file indexes is that they can be easily partitioned. Let us consider a typical distributed web search engine: the index can be distributed across the different nodes.
1.2 Sketching the Architecture of a Web Search Engine

of the underlying architecture in order to enhance the overall system’s throughput (i.e., the number of queries answered per each second). For this purpose, two different partitioning strategies can be devised.

The first approach requires to horizontally partition the whole inverted index with respect to the lexicon, so that each index server stores the inverted lists associated with only a subset of the index terms. This method is also known as term partitioning or global inverted files. The other approach, known as document partitioning or local inverted files, requires that each index server becomes responsible for a disjoint subset of the whole document collection (vertical partitioning of the inverted index). Figure 1.5 graphically depicts such partitioning schemes.

The construction of a document-partitioned inverted index is a two-staged process. In the first stage each index partition is built locally
and independently from a partition of the whole collection. The second phase collects global statistics computed over the whole inverted index. One of the most valuable advantages of document partitioning is the possibility of easily performing updates. In fact, new documents may simply be inserted into a new partition to independently index separately from the others.

Since the advent of web search engines, a large number of papers have been published describing different architectures for search engines, and search engine components. Many other papers enumerate the major challenges search engine developers must address in order to improve their ability to help users in finding information they need. Interested readers shall find in the above referenced papers many interesting insights. Needless to say, you shall not find any particular details, in this survey, about the real structure of a search engine. Usually, this kind of information is highly confidential and it is very unlikely that search companies will ever disclose them.

1.3 Fun Facts about Queries

Due to their “commercial importance”, finding query logs has always been a difficult task. The very first publicly available query log dates
back to 1997. Doug Cutting, representing Excite, a major search service to that date, made available for research a set of user queries as submitted to Excite. Since then, the other query logs made publicly available were the AltaVista log, the TodoBR query log, and the AOL log.

AOL eventually fired employees involved in the public release of their log. This confirms, even more strongly, the particular level of privacy characterizing such data. Obviously, this may sound worse than it is. Search Engine companies are still releasing their data, only that they adopt more conservative policies and release data under research licenses preventing broad distribution.

Figure 1.5 shows a cloud of the 250 most frequent queried terms in the AOL query log.

Queries posed by users are somewhat entertaining. To have an idea of what every day users search through search engines, consider these queries that were actually extracted from the (in)famous AOL Query Log.[1]

In today’s hectic world, people often get very stressed. Stress produces distraction and user #427326 probably was a little more stressed than the average. At 2006-04-21 21:16:51, in fact, he was looking for the following sentence “where is my computer”. Well, probably is closer than what you were suspecting. Actually, searching for this sentence on popular search engines result in around 200,000 results. Gosh! Many stressed people out there.[2]

Again, people gets stressed easily today. I dare you to guess what was user #582088 looking for by entering the following keywords “can you hear me out there i can hear you i got you i can hear you over i really feel strange i wanna wish for something new this is the scariest thing ive ever done in my life who do we think we are angels and airwaves im gonna count down till 10 52 i can”. Hint: try by yourself and enter the above sentence. What is the result? In your opinion, what was user doing while typing the query?

Search engines publish some of the most interesting submitted queries. Interestingness, here, is a relative concept. Depending on the


[2]Indeed, many results are on people asking where is “My Computer” icon on their desktop.
Introduction

search engine company, interesting may mean different things. At Google, for instance, Zeitgeist\(^3\) is a

> "cumulative snapshot of interesting queries people are asking over time, within country domains, and some on Google.com that perhaps reveal a bit of the human condition."

Zeitgeist does not reveal the most searched queries, but only those having had a “sudden”, and “unexpected” raise in popularity. For instance, late in 2007 Italian Zeitgeist ranked “federico calzolari”\(^4\) as the most “inflated” query. Many (mainly Italian) newspapers, and blogs started to ask who is the person referred to in the query. The name was that of a Ph.D. student in Pisa that periodically queried Google for his name. This resulted in an unexpected raise in popularity for the query term thus ending up in the Zeitgeist. Many people, mainly journalists, started to discuss whether or not Federico Calzolari has hacked the Google ranking algorithm.

It is important to point out that the discussion above seems to imply that one could guess the intent of the users by looking at query session. This is far from being true. As it is shown later on, the identification of users’ tasks is a very challenging activity. The main goal of this paragraph is to make readers aware of: (i) the variety of information in query logs, and (ii) the detail that, in principle, can be obtained about a single user.

An interesting recent paper dealing in a scientific way with discovering information about search engine index content by carefully probing it using queries out of a query log is Bar-Yossef and Gurevich \(^28\).

1.4 Privacy Issues in Query Log Mining

The most recent scandal concerning privacy and query logs happened in 2006 at AOL. AOL compiled a statistical sampling of more than 20 million queries entered by more than 650,000 of their customers, and then made this DB available to the public for research purposes.

\(^3\)http://www.google.com/press/zeitgeist.html.
1.4 Privacy Issues in Query Log Mining

While user names were replaced by numbers, these numbers provide a thread by which queries by a given user could be identified so that if, for example, a user entered some piece of information which permits their identity to be discerned, all the other queries they made during the sampling period could be identified as theirs. AOL received so much criticism for releasing this data that it eventually fired two employees. The real problem was that they released ALL off the data to EVERY-ONE. A Non-Disclosure-Agreement form for researchers to sign, would have saved a lot of pain to AOL people that were fired after the mishap.

Many commercial search engines overcome to this problem by simply not publishing their logs. Is this approach good? Yes for some reasons, no for others. Roughly speaking, it is good that people (in general) cannot access query log data. As already said above they might be used to infer users’ preferences, tastes, and other personal information that might be used against their will. On the other hand, as pointed also out by Judit Bar-Ilan in [27]

“[…] interesting results can be obtained from query logs without jeopardizing the privacy of the users.”

While Bar-Ilan showed that it is possible to sanitize a query log in order to prevent private information to be disclosed, Jones et al. [117] showed that even heavily scrubbed query logs, still containing session information, have significant privacy risks.

This paper does not deal with this (extremely important) issue, but we would not have been comfortable without making the reader aware of this issues. More important, we think this would clarify why many studies reported here are made on (sometimes) old and outdated logs, or logs privately held by companies not sharing them.

The interested reader shall find an introduction and some thoughts about privacy and log publishing in recently published papers [1] [126] [164] [230]. Recently, Cooper published a very detailed survey on query log privacy-enhancing techniques [64], readers interested in this topic shall find a very thorough analysis of the most recent techniques dealing with privacy preserving analysis of query logs.
Recently ASK\textsuperscript{5} has given the possibility to users to explicitly deny the storing of their usage data. On the other hand, Google, Yahoo, and Microsoft, continuously ask users for the permission to store their preferences, behaviors, and data in general. What is the most correct behavior? It depends on search engines’ policies, thus we do not enter into details on how these are managed.

The remainder of this work presents the most recent results and advances that have used query logs as (the main) source of information. It is worth mentioning here that not always the experiments presented might be reproduced. This is something that in science should be avoided \cite{87}. Unfortunately, as already said above, the main source of knowledge (the query logs) are mainly kept by search engine companies that for many reasons (not last, privacy issues) are very reluctant of give them away, even to scientists. Therefore, many times in this article, the experimental evaluation is based on results obtained by others and presented in the literature. We apologize in advance to both authors of the mentioned papers, and to readers.

Before entering into the details of our survey, it is important to remark that query log mining is a very hot topic nowadays. The material covered by this survey is to be considered as a valid starting point for those interested in knowing something more on the topic. Proceedings of the major conference series (e.g. SIGIR, WWW, SIGMOD, VLDB, SIGKDD, CIKM, etc. Just to name a few) and top journals (e.g. ACM TOIS, ACM TWEB, ACM TKDD, ACM TOIT, Information Processing & Management, JASIST, IEEE TKDE, etc.) are the best source for the state-of-the-art works on this field. Furthermore, we use the same notation used by the authors of the surveyed papers. This, in our opinion, makes each (sub)section of the survey more independent and leave to the reader the possibility of selecting the techniques he is interested on.

That said, let the journey into the marvelous world of queries begin…

\textsuperscript{5}http://www.ask.com.
References


References


References


References


References


Full text available at: http://dx.doi.org/10.1561/1500000013
References


References


References


Full text available at: http://dx.doi.org/10.1561/1500000013


References


References


References


Full text available at: http://dx.doi.org/10.1561/1500000013
References


References


References

175


References


