How Search Engines Work

Andrew O. Loutfi
Luther College
loutan01@luther.edu

Abstract

This paper provides an overview of how the hypertextual search engine Google works, how Google compares to other well known hypertextual search engines, and the future of searching beyond hypertext search engines.

1. Introduction

The World Wide Web is a large collection of trillions of webpages, exabytes of data, and billions of cat pictures. This was also true in a relative sense during the 1990s, when pages were still being measured in hundreds of millions, and data in terabytes. Without the ability to traverse this data however, there is no point for it to exist!

Enter the hypertextual search engine, an instrument of search through which queries are ran against web documents written in HTTP or other similar markup languages, returning a result that aligns most closely with the parameters of the query.

The first search engine was developed by Cornell University Professor Gerard Salton in the 1960s. Salton's search engine algorithm, SAAP (Salton's Magical Automatic Retriever of Text) functions as a high level blueprint for all modern day search engines. The algorithm functions as follows:

1) Take a query and match it against a collection of documents

2) Calculate a list of relevant results and display them by relevancy. (Wright 2016)

Hypertextual search engines can traverse the web autonomously, or with human maintained indexes. One of the search engines that does the latter is Yahoo. Yahoo's approach of having analysts comb through the Internet is simple in theory, but is not without its faults. This is because coverence of topics can be subjective, the system is slow to improve, and expensive to maintain. Many times, services like this return many junk results, through which the user would have to traverse to find what they were looking for. The reason for this is because the number of documents a search engine is able to look for has increased exponentially, while our ability to look for documents has not (Brin and Page 1998).

This model of human maintained index driven hypertextual search engines was the norm throughout the early days of the web. This would change in the late 1990s when two computer science graduate students named Sergey Brin and Lawrence Page developed a new much more automated hypertextual web search engine. The name of this service is Google.

2. How Google Works

When Brin and Page created Google, they outlined three high-level challenges that Google would have to address. The system must be able to download hundreds of gigabytes of web pages quickly–a process known as web crawling–and then index the results so they are search-able. In addition to this, both the indexing system and web crawler must use storage space optimally to maximize efficiency (Brin and Page 1998). In mathematics, googol is the large number 10 to the power of 100. This name was chosen to highlight that the system being built must be infinitely scalable.

Google uses its web crawling programs to constantly crawl the web for links to either new webpages, more up to date versions of webpages, and other documents. The crawler will then parse each page for links, retriev-

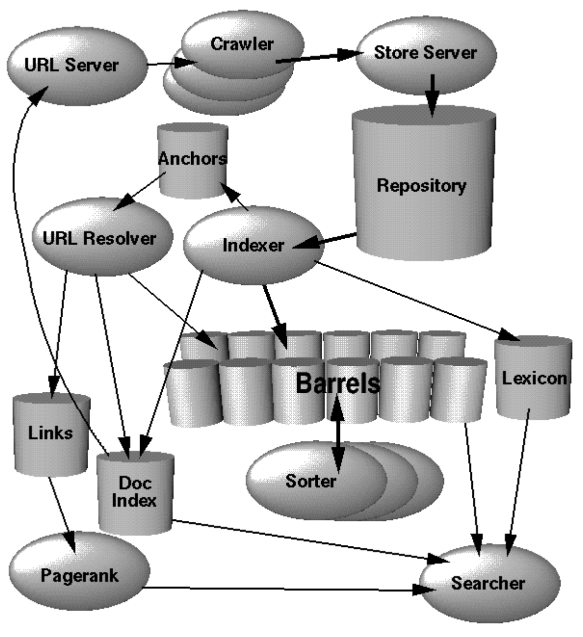


Figure 1. High Level Overview of Google Architecture

ing pages from those links as well. After a web crawler has finished parsing the data it is downloaded to one of Google's Storage Servers and compressed. Each page stored in the Storage Server has a unique ID associated with it. This ID is stored in the document index, where information about each page is kept track of(Brin and Page 1998).

The indexer preforms a variety of functions related to parsing and sorting. After pulling a page from the Storage Server, the indexer will decompress the page, and proceed to parse it. From each document is pulled the occurrences of words, known as hits. These hits are hashed wordIDs by the lexicon where they are logged for fast retrieval and are then distributed into a set of "barrels", creating a forward index(Brin and Page 1998).

These barrels are a hashable datastructure, allowing for elements to be accessed very quickly. The keys are stored in another quick-access type datastructure called a lexicon. If a document contains words that fall into a particular barrel, the docID is recorded into the barrel, followed by the list of wordIDs with hit lists which correspond to those words(Brin and Page 1998).

It should be noted that rather than the words themselves, the wordID of the words are stored in the barrels to optimize space. The forward index is only partially sorted when put into these barrels. This is because the words are stored at a relative difference from the minimum wordID in said barrel(Brin and Page 1998).

As well as the number of hits, the position of the word relative to the document and its font and size is recorded. When the indexer encounters a URL, an association is made with the page the URL is found on as well as the page it is pointing to. All of this information is stored in an anchors file, and is then used to determine where the link points from and to, as well as the text from the link(Brin and Page 1998).

This anchor file is then read by the URLresolver, a data structure which will turn URLs found on the page that have been deemed legitimate into the pages new ID. This means that the contents of a page is directly associated with its URL(Brin and Page 1998).

The sorter takes the barrels, which are sorted by docID, and re-sorts them by wordID to generate the inverted index. This is done in place so that little temporary space is needed for this operation. The sorter also produces a list of wordIDs and offsets into the inverted index. A program called DumpLexicon takes this list together with the lexicon produced by the indexer and generates a new lexicon to be used by the searcher. The searcher is run by a web server and uses the lexicon built by DumpLexicon together with the inverted index and the PageRanks to answer queries(Brin and Page 1998).

PageRank is an algorithm that makes use of the World Wide Web's link structure to calculate a quality metric for each web page. It served as the backbone of the original Google Search algorithm, and is still very much so its bread and butter. Ironically, the algorithm is actually named after Lawrence Page, rather than the service it provides(Brin and Page 1998).

Before Google, no other search engine had used links to structure the web. These links, held in a dictionary-type datastructure known as a map, allow for quick calculation of a pages PageRank. The algorithm also makes use of already published of academically cited literature in order to gauge a cites importance and quality(Brin and Page 1998).

$$PR(A) = (1-d)+d(PR(T1)/C(T1)+...+PR(Tn)/C(Tn))$$

The calculation assumes that a page has t1-tn citations either on the page or pointing to it. The damping factor parameter, d is set between 0 and 1. In their tests of Google Brin and Page have set the damping factors default value to 0.85(Brin and Page 1998).

PageRank corresponds very well to the subjective idea of importance. This makes it a very useful metric when running a search, as very popular results will be prioritized by the algorithm. Furthermore, it means that it can be thought of as a model for user behavior. Assume there is a web surfer who randomly visits web pages by clicking on links, but never presses the back button. They stay on a page until becoming bored of it and start reading another page. The probability that the surfer visits the page is its page rank. The probability that the surfer will get bored and request another page is the damping factor(Brin and Page 1998).

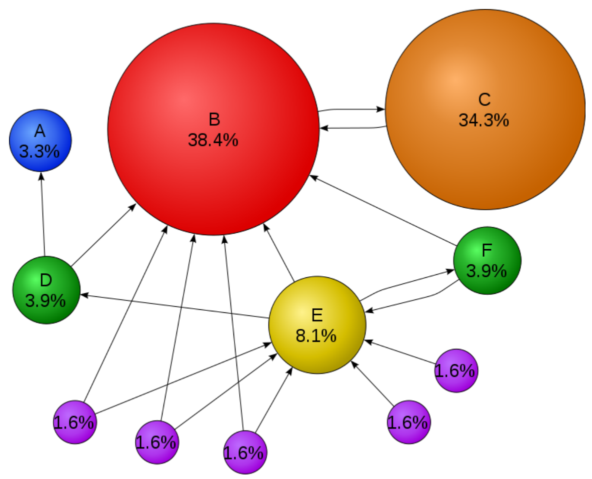


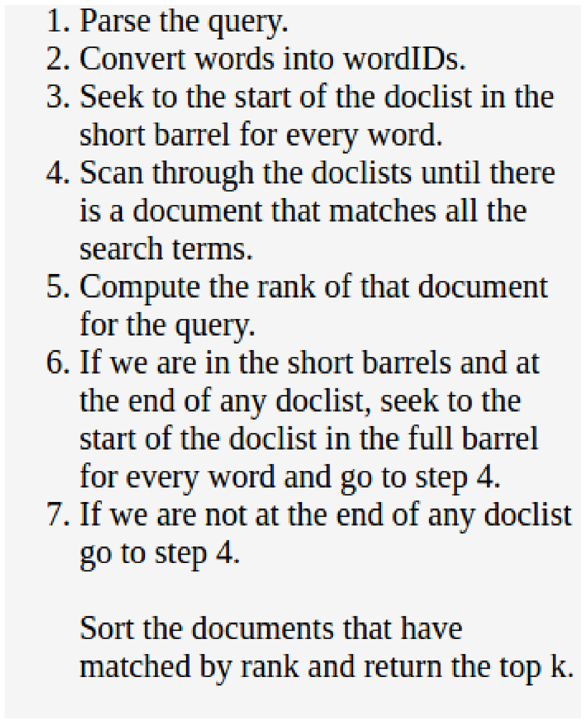
Figure 2. Simple Example of PageRank

PageRank is computed using a logarithmic scale. For the sake of simplicity this example uses percentages to express PageRank. The higher a percentage, the higher the PageRank is. Note that even though page C has a higher PageRank than E that page C only has one direct link in the graph. This one link however comes from Site B, and is therefore much higher in value than pages that have multiple links to one another(Brin and Page 1998). it used to be that you could lookup the page rank for a specific site using Google's Directory. This was discontinued in 2011(Wright 2016).

If web surfers who start on a random page have a likelihood of 0.85 for visiting a random link from that page they are visiting will also have a 0.15 chance of visiting another page at random from the entire web. For example, this means they will hit page B 38.4% of the time. It should be noted that the 0.15 likelihood of jumping to random web page corresponds with the 0.85 likelihood. This is the damping factor(Brin and Page 1998).

Without the damping factor, everyone would end up on pages A, B, and C while all other pages would have a PageRank of zero. In the presence of damping, Page A effectively links to all pages in the web, even though it has no outgoing links of its own(Brin and Page 1998).

When a surfer is searching Google, they are not actually searching the web! Instead, they are searching Google's index of the web. The following excerpt from the Brin/Page paper outlines Google's query algorithm:



1. Parse the query.
2. Convert words into wordIDs.
3. Seek to the start of the doclist in the short barrel for every word.
4. Scan through the doclists until there is a document that matches all the search terms.
5. Compute the rank of that document for the query.
6. If we are in the short barrels and at the end of any doclist, seek to the start of the doclist in the full barrel for every word and go to step 4.
7. If we are not at the end of any doclist go to step 4.

   Sort the documents that have matched by rank and return the top k.

Figure 3. High Level analysis of Google query algorithm

It should be noted that when computing the rank of a document today, Google uses more than 200 different algorithms in the process. Many of these algorithms are classified(Sullivan 2013).

Brin and Page initially designed Google to be scalable up to 100 million web pages. They state that the search runs in linear time, and that its complexity would increase exponentially at over 100 million web pages. This is attributed to nothing more than the limits of the then Solarix/Linux operating systems that Google was running on at the time(Brin and Page 1998). Since then, Google has built data centers with vast amounts of computing power, capable of processing the hundreds of billions of web pages that exist on the World Wide Web today.

3. Search Today: Bringing order to chaos

When Google was created in 1998, it had the ability to download and index 11 million pages in 63 hours, translating to 48.5 pages per second.(Brin and Page 1998) In 2015, Google indexed 60 trillion web pages, translating to roughly 1,902,588 pages per second.(Wright 2016) This massive, uncontrolled expansion of the web presents many exciting opportunities and challenges for Google and other search companies when it comes to making the web indexible.

In 2013 Google updated its search algorithm to make use of natural language processing when parsing a query. This update, named Hummingbird, allows for more meaning to be attached to the query as a whole instead of each particular element of it. The goal is that search results are further curated by the queries meaning, meaning they will be more relevant(Sullivan 2013).

Google's search algorithm was updated again in 2016 to include a machine-learning reliant data structure called RankBrain. This algorithm uses trigger words and phrases it has learned through parsing queries of many users to return results that are relevant, but may not have the exact words being searched for(Sullivan 2016).

One final way that Google has adapted its search algorithm is to treat common phrases the same way it does words, hashing them into barrels and extracting them with a phraseID(Wright 2016).

Along with natural language processing and new features in markup languages, this has helped give rise to some powerful and intuitive features of Google's rich search result ecosystem.

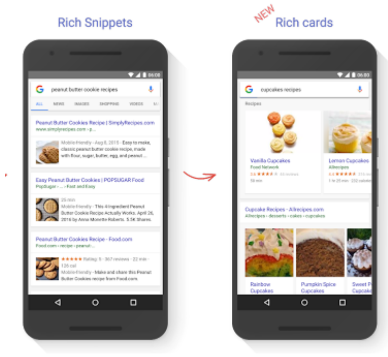


Figure 4. Mobile view of rich snippets and cards

Google's rich search results can be found in the form of snippets, cards, and quick answer boxes. With rich snippets, web developers of sites with structured content such as business listings, news articles, reviews, and cooking recipes can markup their content so it can be displayed within the search results page. Since there creation, rich snippets have given rise to rich cards.

While these features are geared towards mobile devices, it can certainly be argued that both rich snippets and rich cards are just as engaging when viewed on a personal computer, especially when compared to traditional search results. Both of these search data structures make use of structured data formats to display content in a more visual and engaging format. This structured data format, PageMap, is what enables web developers to programmatically embed data right into the search results page(Zohary 2016).

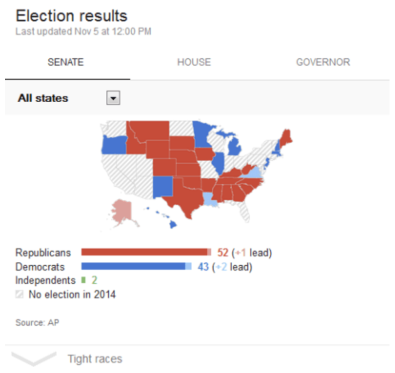


Figure 5. Quick Answer Box displaying Election Results

Quick answer boxes also provide an intuitive and engaging way of displaying search results. In election cycles for example, Google's search team will create a quick answer box that provides real-time election updates and display them with visuals. They are also used when looking up words, people, and can be used to specify download links that have been cleared by Google to not be malicious. This gives rise to another issue of expansion that Google must deal with regarding search: content quality.

During the Fall 2016, a number of fake news articles were propagated through major websites such as Google and Facebook regarding the presidential candidates from both parties. It is speculated that these articles affected the outcome of the presidential election. Whether or not this is true, it is something that must be dealt with, as one of the responsibilities of a search engine is to deliver accurate and credible information. Google has began to deal with this by withholding add revenue from websites with fake news on them, but greater initiatives are required.

Google could remove content manually, but just like human maintained index driven hypertextual search engines, it is far from efficient and much more subjective when compared to curating content algorithmically.

Traditionally, Google has dealt with news the same way it does the rest of the web. It displays news links from sites deemed by its ranking algorithms to be the most authoritative: the most commonly viewed site. This means that the most major part of the determinants of a news story's ranking is popularity instead of legitimacy. This means that if a fake news article is popular, it is more likely to go viral(Bowden 2016)!

Google's ability to algorithmically determine facts and truth is still an up and coming technology. That being said, Google should not be the only entity able to define whether something is fact. Having the ability to both define truth and present it to society concentrates a lot of power that could be easily exploited(Bowden 2016). Google has recognized this, as it actively funds algorithmic fact checking initiatives around the world, as well as its own research.

With the advancement of up and coming technological trends such as natural language processing, location-aware devices, voice recognition, and the never ending stream of new features and programming languages software engineers and data scientists have made great strides in the quantification of the World Wide Web.

References

I. Bowden. How google is tackling fake news, and why it should not do it alone, 2016. URL `http://searchengineland.com/google-tackling-fake-news-not-alone-264058`.

S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine, 1998.

D. Sullivan. Faq: All about the new google hummingbird algorithm, 2013. URL `http://searchengineland.com/google-hummingbird-172816`.

D. Sullivan. All about the google rankbrain algorithm, 2016. URL `http://searchengineland.com/faq-all-about-the-new-google-rankbrain-algorithm-234440`.

A. Wright. Re-imagining search, 2016. URL `http://cacm.acm.org/magazines/2016/6/202656-reimagining-search/fulltext`.

N. Zohary. Introducing rich cards, 2016. URL `https://webmasters.googleblog.com/2016/05/introducing-rich-cards.html`.