



Real-Time Search

Discovering the Web in Real-Time

The Web has become the principal medium through which users access an ever-growing wealth of information. The world is advancing at an accelerated rate, yet the contemporary means of search and discovery, search engines, remain stagnant. They're lacking in their ability to deliver the Web to users in a timely fashion. Now, real-time search has emerged as the next generation, reducing the lags and delays in accessing quality information from the entire web. The problem with most real-time search has to do with comprehensiveness: in order to achieve fast performance, existing real-time search systems are limited to an extremely small set of web sites. The tension between real-time response and comprehensive coverage requires a new way of thinking about the entire architecture of search.

Background

Indexing is a core part of modern search and Information Retrieval (IR) systems. Current approaches are still dominated by techniques developed a long time ago, for systems with hardware requirements very different from today. Real-time search presents a completely different set of challenges requiring a completely different approach to indexing. The challenges are especially difficult because the scope is still enormous – the entire web! – and the user’s expectations are for information being indexed as fast as it appears on the web, with a lag measured in seconds and (fractions of) minutes.

The issue of scope, or scale of the problem is very important. It used to be the case that one could index a very large corpus of data with a significant lag and expense of index creation (e.g. traditional search engines), or index a small corpus of data really fast (e.g. online news), but not both.

Our goal is to index the data appearing on the entire web, in real-time, with no compromises in quality or user experience.

One of the principal data structures in an index is a *termlist*. This is essentially a list of (occurrences or IDs of) documents in which a given term (keyword) appears. Simply put, an index can be thought of as a very large collection of termlists, with a fast and efficient method for their lookup and retrieval.

The principal hardware requirement that drove index design is the primary difference in the latency of disks (rotating magnetic media), and Random Access Memory (RAM). Modern disks have access latencies in the range of 10 ms while RAM access, even in cases of processor cache “misses”, is on the order of 10 ns. That’s a difference of 6 orders of magnitude!

Note that historically there has been an implicit assumption that it is not possible to fit the entire index in RAM. This assumption is increasingly untrue and we will discuss its important ramifications in later sections.

The slow access time of disks and the necessity of using them resulted in architectures optimized for avoiding disk head seeks, with the result of storing index data (termlists) in consecutive sequences of bytes that are as long as possible. Ideally, we would want every termlist to be stored in a single consecutive sequence on disk.

The easiest way to store data under such a scheme was to wait for a corpus of documents to be large enough in size and then create an index where all termlists are consecutive by construction. This process of index construction was very expensive in terms of time, processing and data transfer, but since it could be done infrequently the cost could be amortized. This is how batch indexing was born.

Note that first search engines had update cycles on the order of months! Even Google, until about 2003, had update cycles on the order of weeks. They have improved the lag for some specific types of content but the update cycle for the bulk of Google's index is still too long.

Another reason for the large cost of indexing was the advent of link-based ranking techniques such as Google's PageRank. Such techniques have significant processing costs since they require computation over the entire (sparse) Web link-graph.

The Real-Time Search Problem

Real-Time Search has recently emerged as a new and exciting trend in search. It places far greater emphasis on freshness and on minimizing the lag between acquisition of content and its appearance in the index. Real-Time Search completes this process on the order of (fractions of) minutes, possibly even seconds.

A very significant portion of search engine queries (40%) are made by users looking for the latest and freshest information. In addition, users expect these freshest results to be of the highest quality. Real-Time Search greatly improves the user experience for those queries seeking the freshest, highest-quality results.

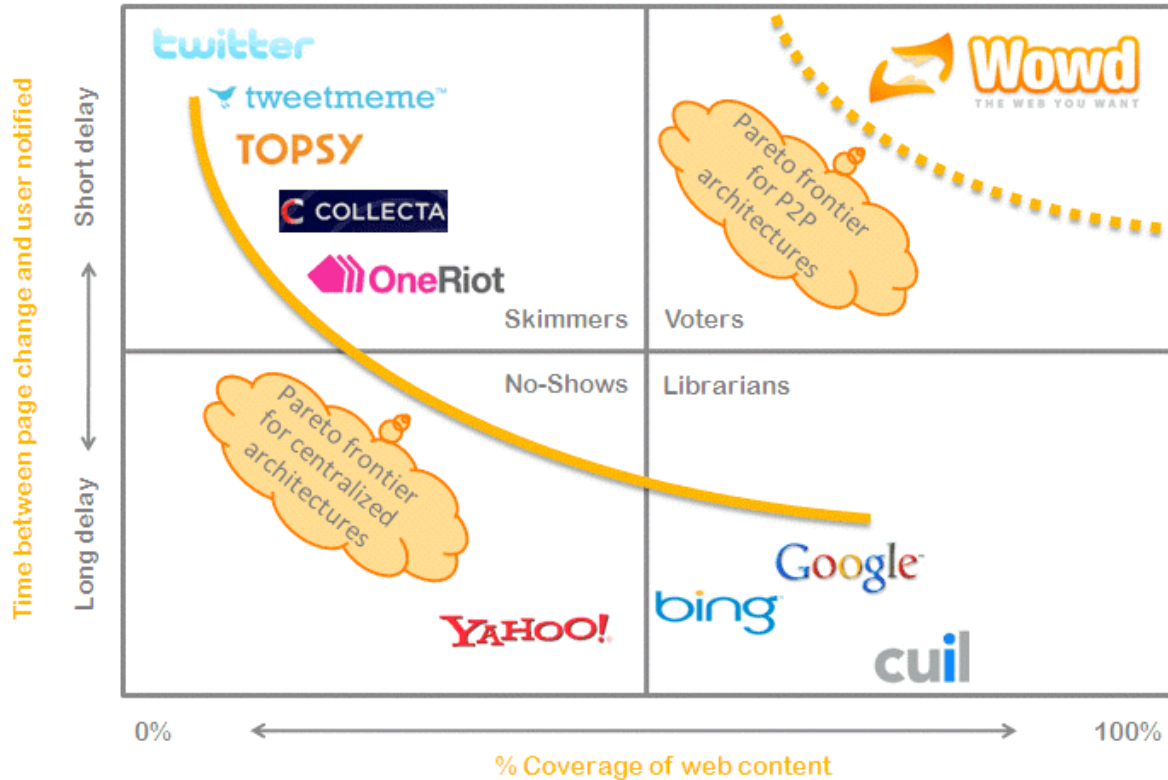
As previously explained, traditional index architectures have been optimized in almost the exact opposite way, with large lag times for the appearance of fresh links in results in order to amortize the very high cost of construction of the entire index.

Current real-time search architectures, for instance, Twitter Search, place an almost exclusive emphasis on freshness and inclusion of *all* results matching a given keyword. Such an approach sounds appealing but it poses major problems for keywords which appear frequently in documents. The resulting stream of matching results is very large, changes (too) quickly, and most of the results that it contains are of low quality and are not particularly interesting.

This is why the notion of *quality* is of paramount importance in real-time search as well as general search. We will discuss quality and ranking in more detail in subsequent sections.

Current Search Marketplace

Two of the key aspects of the search problem are coverage of the Web content and the lag between discovery of content and its availability in the results. Contemporary general search engines such as Google, Bing and Yahoo have been focused on *coverage*, trying to maximize the scope of the Web coverage by increasing the size of the index. On the other hand, the newly emerging players in the field of real-time search have been focused on reducing the lag in displaying the results to users. This situation is illustrated in Fig.1



The X-axis represents the coverage of the Web while the Y-axis represents the lag between page discovery and its potential inclusion in search results. The general search engines are grouped in the lower right quadrant, with large coverage of the Web but also with a long delay in including new material. On the other hand, current real-time search players are in the upper-left quadrant, with low lag but with extremely limited coverage of the Web.

Wowd is in the upper-right quadrant, with both low lag in the inclusion of new content and offering indexed coverage of the entire web.

Note that the tradeoff between lag and coverage in centralized architectures is represented by the curve labeled “Pareto frontier of centralized architectures”. It simply represents the limits in the space of tradeoffs between lag and coverage.

The Pareto frontier of distributed architectures illustrates that the space of tradeoffs in the distributed space is quite different, due to intrinsic superior scalability.

Real-Time Search Intensity

We have seen that the need for efficient use of disk resources is of great importance in controlling the cost of indexing and how the real-time search problem has almost opposite requirements. It should not be surprising then that

real-time search is very resource intensive and costly. The cost of indexing cannot be amortized at all since the index is constantly changing and the system must support continuous changes.

The simplest solution is to avoid use of disk altogether and rely on storing the entire index in RAM. This approach has a severe cost limitation in the amount of RAM that can be deployed at a reasonable cost. In addition it is exacerbated by reliance on available existing indexing systems (e.g. Relational Database Management Systems, Lucene) that have been optimized for disk-based systems, which do not function optimally in pure RAM environments. This is why we believe that new index architectures have to be created to address the requirements of real-time search.

Real-time search is an instance of general search and as such is subject to the scale of the entire Web and the rate of change of content on the Web. This is why *scalability* is of paramount importance. Scalability will be covered in more detail in subsequent sections.

Sliding Window

One prominent feature of current real-time systems, that so far has not received much attention, is the *sliding-window* nature of their indexing. By this we mean the notion that only results within a specified fixed time interval are indexed and as new results appear, the old ones are discarded.

We strongly believe that this approach is fundamentally flawed as it is only able to return quality results within a short, fixed time-window and only for a limited fraction of content. There are many topics, events, discussions and quality results that fall outside of any fixed-size window. Indeed, we would argue that the span of *most* topics of interest fall outside any fixed-size window, and that as a result, the current approach to real-time search is essentially broken.

Our Solution

Our solution for real-time search has been designed from scratch to be completely scalable in order to tackle the enormous scope of the Web and the rate of change of its content. We achieve scalability by efficiently leveraging enormous resources available in distributed systems with multitudes of users. Each individual user's contribution is small and imperceptible from their perspective but in aggregate, the available resources are enormous.

Consider RAM as an example: modern operating systems and even browsers, which can be increasingly thought of as a sort of Web operating system, require significant amounts of RAM. In addition the cost of RAM has been dropping precipitously. Assuming an application using 100MB of RAM and a collection of, say, 1 million users, the total amount of RAM available in the system would be 100 Terabytes! Such a quantity of RAM is comparable to contemporary general search

indexes for the entire Web deployed by Google, Yahoo and Microsoft. Note the prohibitive cost of a centralized system of such magnitude: assuming \$150/GB cost of RAM together with machines required to place the RAM, just hardware alone would cost \$15M. Of course, there would be the added cost of datacenters to host the hardware, bandwidth, power, maintenance, *etc.*

Awareness around the issue of cost has been on the rise. A common misconception has emerged that search is the province of only the largest companies which have the resources to cover these massive costs. We aim to demonstrate that this is not the case. In fact, the aggregate power of a more efficient distributed system with millions or tens of millions of users is superior to a centralized system of equal size.

We have already mentioned that an exclusive focus on freshness alone, with little or no regard to quality, can be very detrimental to the quality of user experience with real-time search. Our approach emphasizes both freshness and quality by continuously performing ranking.

Our ranking is based on multiple factors including link analysis, popularity, a multitude of search signals like keywords in the title, as well as other signals like the number of retweets on any given tweet, in Twitter.

Naturally, freshness has a very large bias and is one of the most important factors in ranking. But the key is that freshness is not the only consideration – it is one of many.

Another essential feature of our approach is preservation of older results. Of course, the ranking weight of older results decreases with time but their accessibility is still significant for many queries where there is a lack of quality fresh results or, more specifically, a lack of fresh results from any kind of fixed sliding-window.

There has not been much discussion about the relationship between general and real-time search. We believe there is a very strong, even formal relationship between them. In simple terms, a real-time search engine that preserves older results, with proper time-based biasing in rankings would, in time, start to converge to a general search system. The resulting general search engine would be of the highest quality, because of its unprecedented level of freshness.

This relationship can be expressed by the following formula:

$$Search(t) = \int_{t_0}^t RealTimeSearch(t) dt$$

General search and real-time search are expressed as two functions of time t : $Search(t)$ and $RealTimeSearch(t)$. The starting point of the integration t_0 can be thought of as an arbitrary point in time when the process of integration starts.

In simple terms, the integration can be viewed as preservation and accumulation of older results, along with the preservation of ranking information.

Note that the above formula has a flip side, which can be expressed as

$$\mathbf{RealTimeSearch}(t) = d\mathbf{Search}(t)/dt$$

Which means that real-time search is simply the first derivative of general search – or an instance of general search over some short time-window (dt) which can be viewed as the sliding-window we discussed earlier.

In summary, our approach to real-time search emphasizes results across the entire web, not only a limited set of social networking sites (even though some of those sites, such as Twitter, are of particular importance).

In addition, Wowd does not discard older results, instead it keeps and indexes them together with the newly indexed information (naturally weighted by freshness). With this approach, Wowd is able to return quality results for all queries, not only the ones related to the latest news and events.

Using a scalable approach of indexing the entire Web in the real-time, together with the preservation of all results in the index, Wowd has created a superior quality experience where users are always able to find the latest, quality information they're looking for.