Web Usage Mining in Search Engines*

Ricardo Baeza-Yates

Center for Web Research
Department of Computer Science
Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
rbaeza@dcc.uchile.cl

Abstract

Search engine logs not only keep navigation information, but also the queries made by their users. In particular, queries to a search engine follow a power-law distribution, which is far from uniform. Queries and related clicks can be used to improve the search engine itself in different aspects: user interface, index performance, and answer ranking. In this chapter we present some of the main ideas proposed in query mining and we show a few examples based on real data from a search engine focused in the Chilean Web.

1 Introduction

Given the rate of growth of the Web, scalability of search engines is a key issue, as the amount of hardware and network resources needed is large, and expensive. In addition, search engines are popular tools, so they have heavy constraints on query answer time. So, the efficient use of resources can improve both scalability and answer time. One tool to achieve these goals is Web mining. In this chapter we focus on Web usage mining, which includes logs with queries in addition to clicks, and not in other kinds of Web mining such as link analysys (see [8]), content mining, or Web dynamics (see [13]).

There are few papers that deal with the use of query logs to improve search engines, because this information is usually not disclosed. The exceptions deal with strategies for caching the index and/or the answers [26, 16, 14] and query clustering using click-through data associated to queries (obtaining a bipartite graph) for ranking or related goals [10, 7, 24, 28, 27]. Other papers are focused in user behavior while searching, for example detecting the differences among new and expert users or correlating user clicks with Web structure [11, 2, 15]. Recently, there has been some work on finding queries related to a website [9] and weight different words in the query to improve ranking [18].

The main goal of this chapter is to show how valuable is to perform log query mining, by presenting several different applications of this idea combined with standard usage mining. Although past research has focused in technical aspects of search engines, analyzing queries has a broader impact in Web search and design in two different aspects: *Web findability* and *information scent*. Web findability[1] or ubiquity is a measure of how easy is to find a Web site, where search engines are the main access tools. To improve findability there are several techniques, and one of them is using query log analysis of Web site search to

*To appear in *Web Mining: Applications and Techniques, Anthony Scime (editor), Idea Group Publishing*.

[1]We have found Web pages using this term with our semantic as far as 1995.

use in the text the most used query words. Information scent [17] is how good is a word with respect of words with the same semantics. For example, polysemic words (words with multiple meanings) may have less information scent. The most common queries are usually the ones with more information scent, so analyzing Web search site queries we can find: words that are found (or not found) in a site but have more or similar information scent than words in the home page (which have to be replaced and added); and words that are not found that imply new information to be added.

The chapter is organized as follows. We start with some basic concepts followed by the main statistics of search engine usage. Next we present two applications of the search engine log. First, using the query distribution, we present an inverted file organization that has three levels: precomputed answers, main, and secondary memory indexes. We show that by using half the index in main memory we can answer 80% of the queries, and that using a small number of precomputed answers we can improve the query answer time on at least 7% [5]. Second, we present an algorithm that uses queries and clicks to improve ranking [28], which captures semantic relations of queries and Web pages. We conclude with some open problems.

2 Preliminaries

2.1 Zipf's Law

The Zipf's law was introduced in the late 40's to describe several empirical observations such as the distribution of population of cities or the frequency of words in English written text [29]. If F_i is the frequency of the i-th most frequent event, we have that $F_i \sim \frac{1}{i^\alpha}$ where α is a constant, and the parameter of the distribution. In a log-log graph, α is the slope (without the sign) of the line. In the case of words, it means that there are few very frequent words (usually called stop words) and many unusual words. In figure 1, the first part of the curve are the frequent words, while the last part of the curve are the unusual words. Perhaps, due to this distribution, the number of distinct words in a text (vocabulary) does not grow linearly, but follows a sublinear curve of the form $N = O(T^\beta)$ (Heaps' law), with T the total number of words, and β around 0.5 for English text [1] and a bit larger for the Web (around 0.6).

2.2 Inverted Files

The main components of an inverted file are the vocabulary or list of unique words and the posting file which holds the occurrences in the text of each word in the vocabulary. For each word we basically have a list of pointers to Web pages plus some additional information that is used for ranking the results. The exact implementation depends on the retrieval model used and if we want to use full addressing (for example to answer sentence queries) or just point to Web pages to save space. As the words in the text are not uniform, the posting file has few long lists as well as many more short lists, following the Zipf distribution, which can be compressed efficiently [1].

Due to the Heaps' law, we can assume that the vocabulary always fits in main memory. In practice, the vocabulary of a large search engine has several million of words that will point to hundred millions of Web pages. Due to the constraints imposed by concurrent access to search engines, the inverted file should be in main memory (RAM), but as is too large, part of the index will be in secondary memory (disk), with the subsequent loss in query answer time.

2.3 Web Log Data

For our experimental results we use about 800 thousand pages from a vertical search engine targeted to the Chilean Web, TodoCL [23]. We define as the Chilean Web all .cl domains plus all Web sites with IP provided by Chilean ISP's. In this set of pages we find $T = 151,173,460$ words, from which $N = 2,067,040$ were

unique (vocabulary). The average length of a word in the vocabulary is $\bar{x} = 8.46$. The number of documents where a word appears follow a Zipf's distribution with parameter α about 1.5 as seen in Figure 1 (the model is fitted in the main part of the graph).

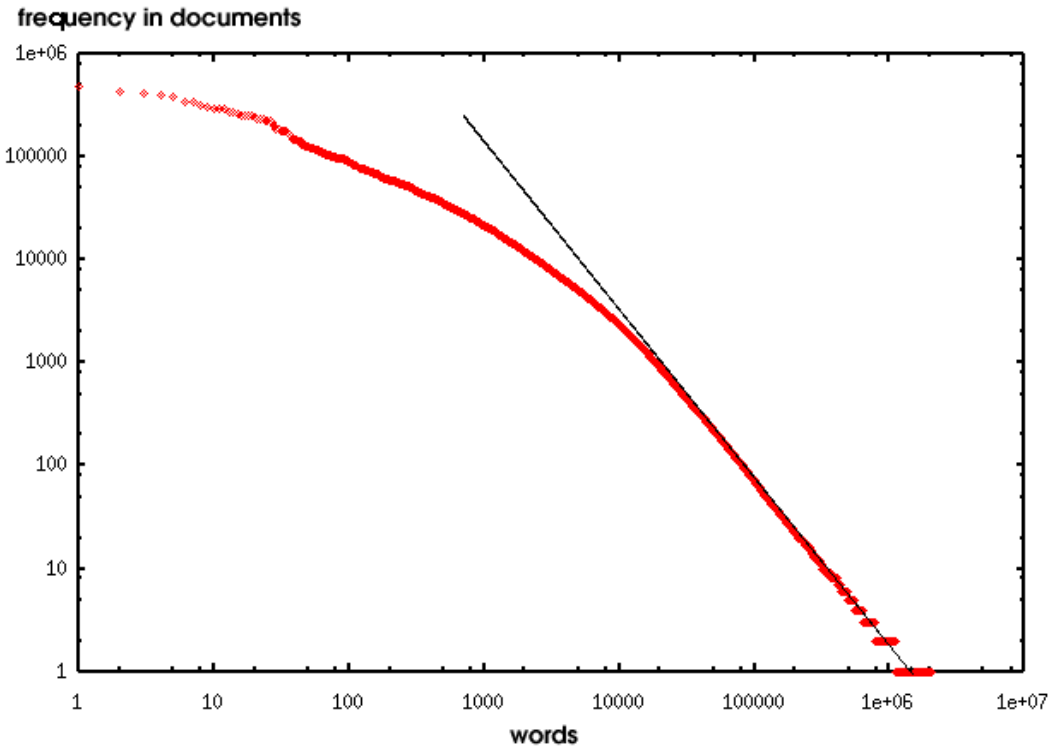

Figure 1: Words in the vocabulary vs. their number of occurrences in Web pages.

The query set, from the same search engine, considers a two month log of 2001 which consists in 777,351 queries containing 738,390 words (that is, an average of 1.05 words per query), where the unique words are $C = 465,021$.

3 Search Engine Usage

In this section we present basic statistics about the queries. Figure 2 shows that the frequency of query words also follow a Zipf's law, with parameter $\alpha = 1.42$. This is larger than the parameter 0.59 obtained in [16], perhaps due to language and cultural differences.

The standard correlation among the frequency of a word in the Web pages and in the queries is 0.15, very low. That is, words in the content of Web pages follow a Zipf's distribution which is very different from the distribution of the words in the query, as is depicted in figure 3 (any correlation would show as a line in the figure.) There are many common words in both the queries and the content, most of them articles, pronouns, etc. (that is *stop words*), but also other words like Chile. Examples of popular queries that do not find many answers are Hentai, Mexico, DivX, covers, and melodies. This implies that what people search is different from what people publish in the Web.

The search engine log also registers the number of answer pages seen and the pages selected after a search. Many people refines que query adding and removing words, but most of them see very few answer pages. Table 1 shows the comparison of four different search engines [19, 25, 21, 22, 6]. Clearly, the default

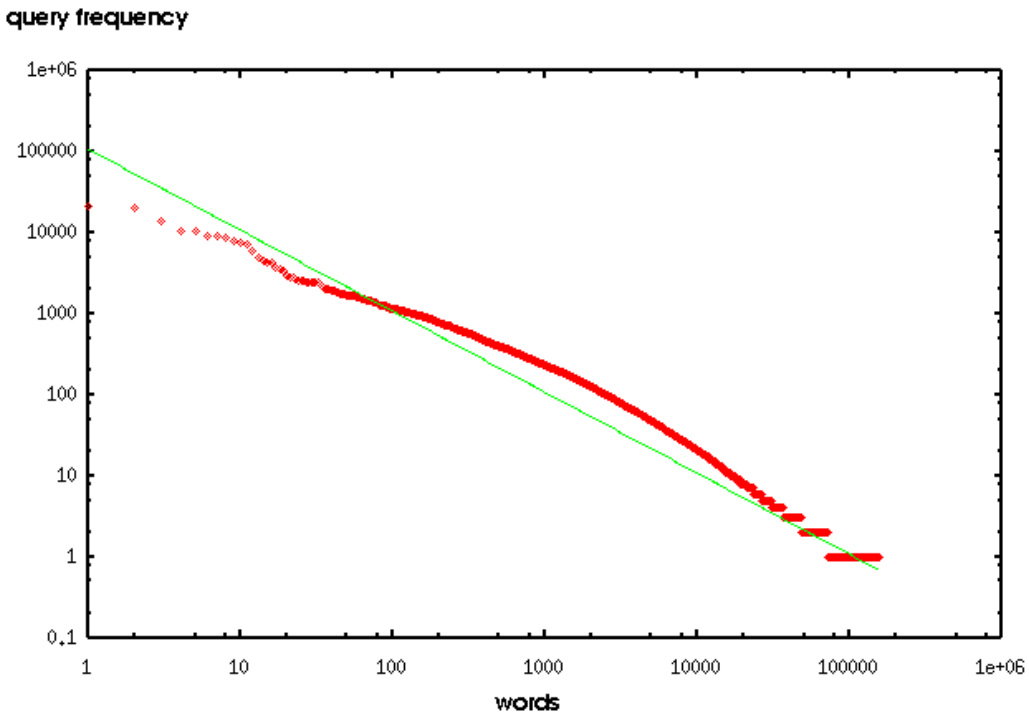

Figure 2: Frequency of query words in a log-log graph.

query operation is dominant (in the case of TodoCL only 15% are phrase queries).

Measure	AltaVista	Excite	Fast	TodoCL
Words per query	2.4	2.6	2.3	1.1
Queries per user	2.0	2.3	2.9	–
Answer pages per query	1.3	1.7	2.2	1.2
Boolean queries	<40%	10%	–	16%

Table 1: Query statistics for four search engines.

The navigation graph can be complicated even in a search engine. Figure 4 shows the transitions between different states, indicating the proportion of users that took that path. The number in each state represents the probability of a user being in that state. From the diagram we can see that:

- Advanced search is not used (but we must have it!).
- Few people refines the query.
- Less than 10% of the users browse the directory[2].

This means that instead of posing a better query, a trial and error method is used. Table 2 gives the most popular queries for three search engines (in the case of TodoCL we have translated them to English), which shows that they are very similar independently of the country and the language. Further studies of queries

[2]TodoCL uses ODP (dmoz.org) as Google.

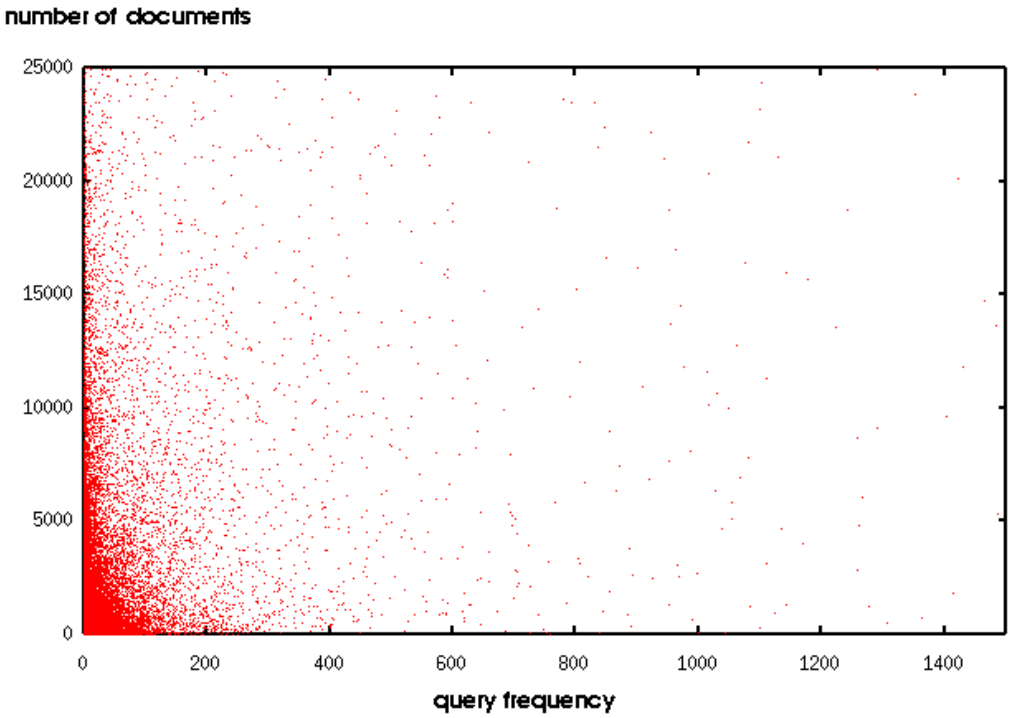

Figure 3: Word query frequency vs. number of documents that contain each word.

have been done for Excite [20, 21] and Fast [22], showing that the focus of the queries has shifted the past years from leisure to e-commerce.

4 Indices based in Query Distribution

In this section we show a simple storage model for a search engine which uses efficiently main memory, improving scalability [5]. As few words are queried frequently, what is important is the growth of the most queried subset of words, rather than the growth of the overall vocabulary on the Web pages. In this section we only use single word queries as they are more than 95% of them.

Storing part of the index in main memory can be considered as static caching of inverted lists, but in practice is not the same because current search engines must have large portions of the index in main memory to achieve fast answer time. In the sequel, we consider that our scheme and caching are orthogonal. That is, we can devote part of the main memory to do dynamic caching of the answers and of the inverted lists that are in secondary memory. Similarly, we could add compression.

4.1 Basic Scheme

Let M be the available main memory. We assume that always the vocabulary fits in main memory (as it grows sublinearly, is a small percentage of the inverted file). So we have that the vocabulary and part of the lists will be in main memory. The rest of the word lists will go to secondary memory.

Recall that N is the number of words of the vocabulary and let L_i be the number of documents where the i-th word appears, where the words $1..N$ are ordered by decreasing query frequency. For each word in the vocabulary we need to store the word itself (let x_k be the length of the k-th word) and a pointer to the

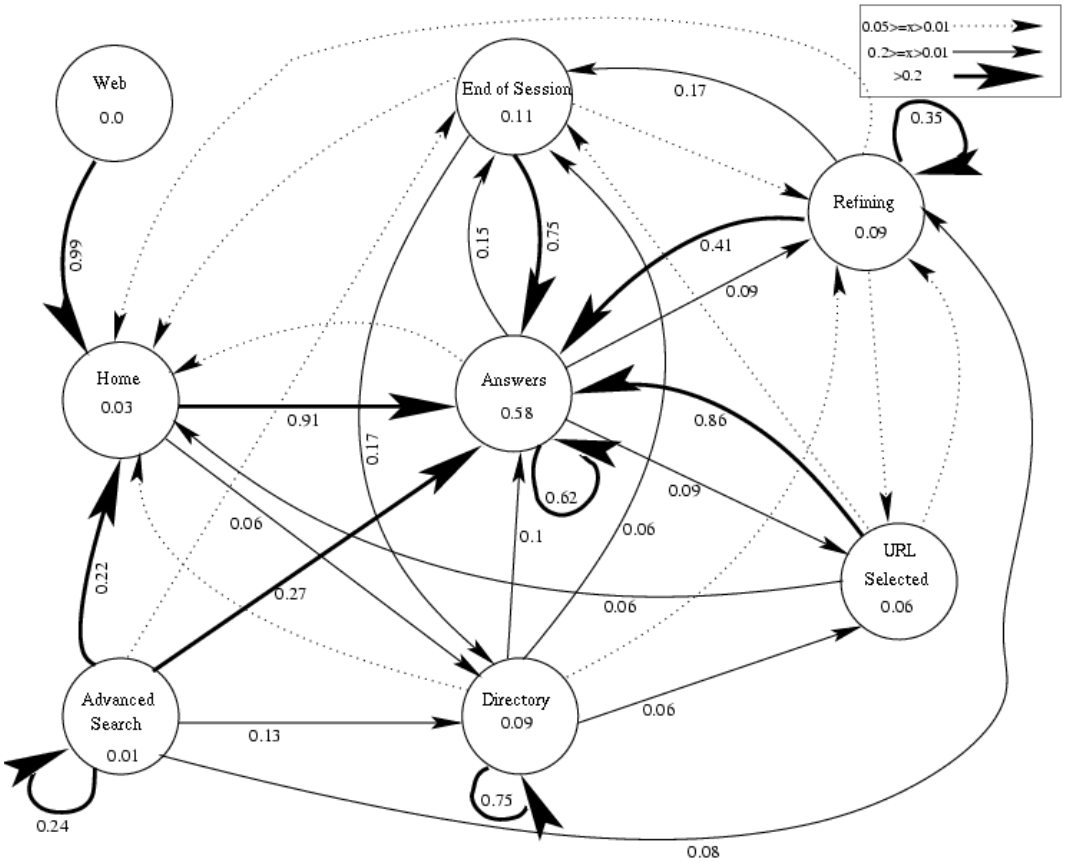

Figure 4: State diagram of user behavior in TodoCL.

corresponding occurrence list (say 4 bytes). In each list element, we need at least an integer to a Web page identifier plus information for ranking purposes (for example, the frequency of the word in the Web page). Let b be the number of bytes per list element, which depends on the exact implementation of the inverted file. Hence, the average space needed by the inverted file, in bytes, is:

$$E = \sum_{k=1}^{N} (4 + x_k + bL_k) = \underbrace{(4 + \overline{x})N}_{V} + b \sum_{k=1}^{N} L_k$$

where $\overline{x}$ is the average length of the vocabulary words, and V the space needed by the vocabulary. Notice that by using $\overline{x}$ we are assuming independence between the word length and the query word frequency. In fact, for our data, the correlation is -0.02, almost nothing.

If $E \leq M$ everything fits in RAM. However, when $E > M$, part of the inverted file must be stored on disk. We could assume that memory paging and disk buffering will improve the access time, but we do not have control of this. Most inverted files use a hashing table for the vocabulary, which implies a random order for the words. If we put in this order the lists of the inverted file until we run out of main memory and the rest on disk, many words that are frequently queried (because the correlation is small) will be in disk. The optimal arrangement would be to store in main memory the subset of query words that maximizes the total frequency and still fits in memory. The rest of the word lists would go to secondary memory. Formally, we have to find the variables $i_1, ..., i_p$ such that they maximize the sum of the query frequencies of the words $i_1, \cdots, i_p$ with the restriction that $V + b \sum_{j=1}^{p} L_{i_j} \leq M$. The optimal solution will depend on each data set.

Excite	Fast	TodoCL
free	free	Chile
sex	download	photos
Christmas	sex	free
nude	pictures	sex
pictures	nude	history
new	mp3	mp3
pics	hotel	videos
music	Windows	music
games	pics	Argentina
stories	crack	Ley
woman	software	university
university	education	sell

Table 2: Most frequent queries in three search engines (2001).

As the distribution is so biased, a nearly optimal solution is to put in main memory the lists of the most frequent query words that fit in it. Let p be the largest number of words in the vocabulary such that

$$V + b \sum_{j=1}^{p} L_j \leq M$$

Then, the lists of the p most queried words will be in main memory. This heuristic is not optimal because there will be a few cases where would be better to replace a word with a large posting list with two or more words with smaller overall lists of similar size or smaller, but at the same time larger total query frequency. Nevertheless, we show next that this solution is quite good.

Using our data, Figure 5 shows the difference between the heuristic (curve below) and a random order (curve above, basically a diagonal). In the x axis we have the portion of query words that have their list in main memory. The y axis shows the portion of the index that must be in main memory to reach that proportion of queries. The vertical line at the end are the words that are never queried. The gap in the diagonal line are two of the most frequently queried words which in our heuristic are at the beginning (the dots of the graphs are plotted in multiples of 100 words). From this example we can see that

- To reach 100% of the queries, we only need to have 80% of the index in main memory. This is because more than 75% of the words are never queried (but have short posting lists).
- To reach 50% of the queries we only need 20% of the index in memory.
- With 50% of the index in memory, we can answer 80% of the queries.

Hence, this simple heuristic reduces the usage of memory in at least 20% without increasing the query answer time.

4.2 Improving the Query Answer Time: Precomputed Answers

Precomputed answers can be considered as static caching of answers, but they are conceptually different because precomputed answers could have better quality, for example with manual intervention (e.g. main links related to a query as in Yahoo!). If we have precomputed answers for the most frequent queries, answer time is improved as we do not have to access the index. However, more memory with respect to the index

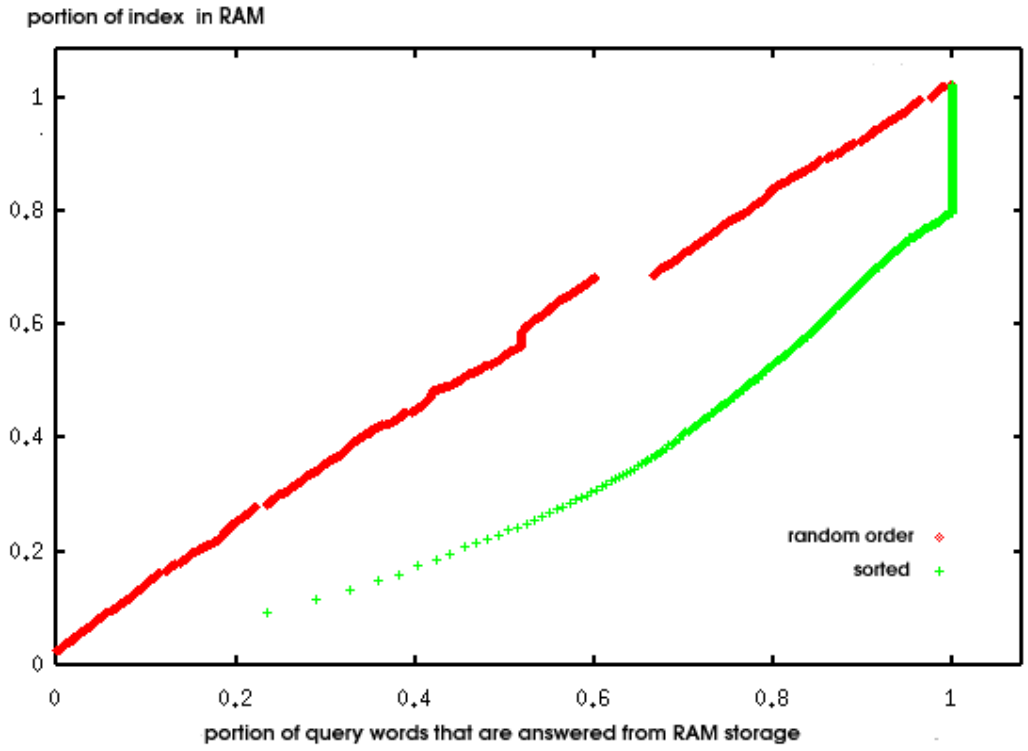

Figure 5: Query frequency vs. memory: random order and our heuristic.

information of these words might be needed, decreasing the value of p and increasing the search time for other words. This implies that there is an optimal number of precomputed answers. We call the precomputed answers a cache memory, although is in main memory.

In our data, with only 2000 precomputed answers we can resolve 20% of the queries. On the other hand, to answer 50% of the queries we need around 100 thousand precomputed answers, which is too much space.

Assume that each precomputed answer needs W bytes of memory (for example, 80 URLs with the text context and other attributes such as size and date). Then we must have:

$$kW + V + b \sum_{i=k+1}^{p+k} L_i \leq M \tag{1}$$

where the words $k+1$ to $p+k$ have their lists stored in main memory. We want to find k and p that optimize the query answer time. Hence we have three sections of memory: precomputed answers, index in memory, and index in disk, that depend on k.

The query answer time is given by

$$t_a = \frac{\beta_1 \sum_{i=1}^{k} F_i + \beta_2 \sum_{i=k+1}^{p+k} F_i + \beta_3 \sum_{i=p+k+1}^{C} F_i}{\sum_{i=1}^{C} F_i} \tag{2}$$

where we recall that C is the number of unique words in the queries and F_i is the query frequency of the i-th word. In addition, β_1, β_2, and β_3 are constants that represent the relative answer time of precomputed answers, index in memory, and index in disk, respectively. In the examples that follow we use the following set of values: $M = 400Mb$, $W = 40Kb$, $b = 8$, $\beta_1 = 1$, $\beta_2 = 5$, and $\beta_3 = 50$. This implies $V = 24.56Mb$,

that is only 6% of the memory. We have chosen W considering 512 bytes per URL, so we have 80 URLs in the precomputed answer. If more answers are needed we have to go to the inverted file in disk and we assume that this event happens with negligible probability. As we have seen in the previous section, the average number of results seen by users is less than 20 [19, 21] and only 12 in our data (but other values for W are possible).

Figure 6 shows the query answer time depending on k (this fixes p), where we can observe that the optimal point is 1338 precomputed answers, improving the answer time in approximately a 7% (the line is the answer time for $k = 0$). Hence, the precomputed answers take 13% of the memory, which implies that in our example the 81% of the memory left is taken by the most frequently queried part of the inverted file.

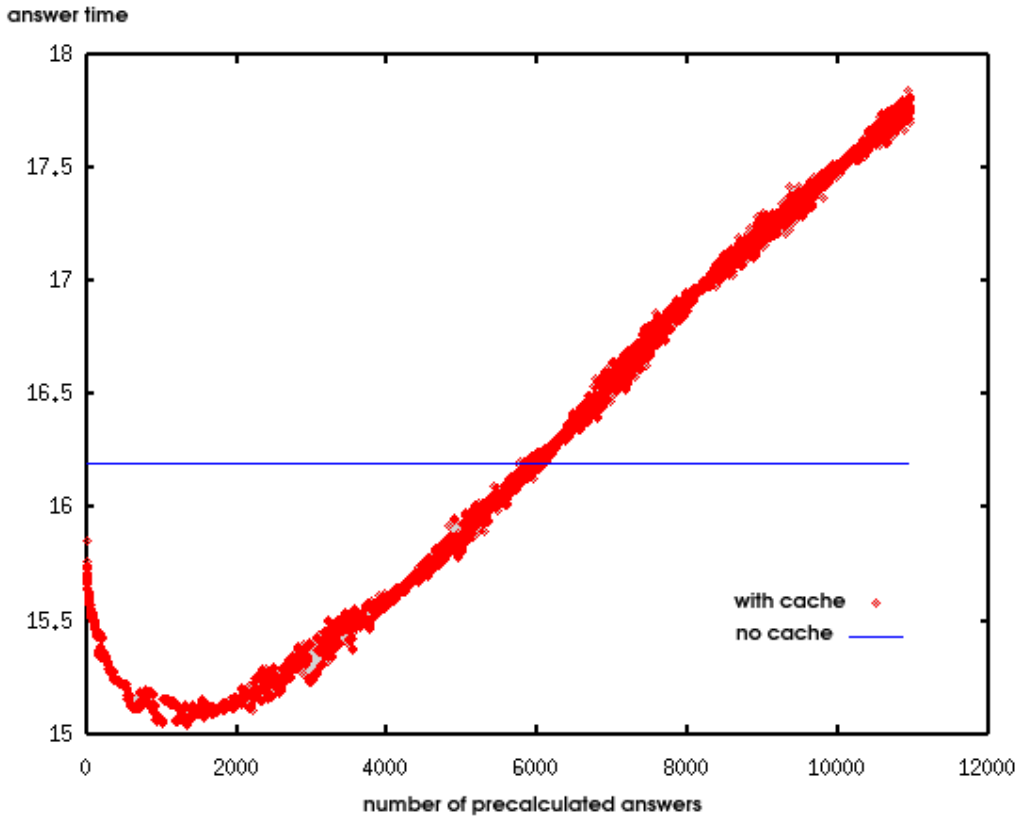

Figure 6: Query answer time in function of the number of precomputed answers.

5 Ranking based on User Queries and Choices

The relation of pages chosen after a query and the query itself gives valuable information. After a query, a user usually performs a click to view one answer page. Each click is considered a positive recommendation of that page (in most cases bad pages are not clicked). In [28] an algorithm based on these relations is proposed to improve ranking: MASEL (Matrix Analysis on Search Engine Log).

5.1 The MASEL Algorithm

The basic idea is to use the query log to find relations among users, queries, and clicks on answers. It is assumed that in a short period of time, behind an IP address there is a single user (not always true because of proxies and dynamic IPs).

The algorithm considers users, queries, and clicked pages, using a recursive definition among those three elements on the spirit of authorities and hubs [12]. The hypothesis is that good users make good queries, good queries return good answers, and good answers are clicked by good users.

Given a query q^*, and a time interval $[t_{now} - T, t_{now}]$, the algorithm has four steps:

- From the query log obtain all users that have asked q^*. We denote by U those users.
- Find from the logs the set Q of all the queries posed by users in U.
- Using the search engine (or other technique), find the set R of relevant clicked pages for all the queries in Q.
- Compute the ranking using the iterative process outlined below.

The period of the log analysis T is a parameter of the algorithm. Let $m = |U|$, $s = |R|$, and $n = |Q|$. We define u_i as a variable that represents the quality of the user i, q_j as the quality of query j, and r_k as the quality of the page k. We use normalized values, that is

$$\sum_U u_i = 1, \quad \sum_Q q_j = 1, \quad \text{and} \quad \sum_R r_k = 1 \,,$$

although we are interested in the order of them. We compute these values iteratively (normalizing them after each iteration). We define the users depending on their queries:

$$u_i = \sum_{j=1}^{n} a_{ij} q_j, \quad \text{where } a_{ij} = \begin{cases} num(i,j) & \text{if } j = q^* \\ \alpha num(i,j) & \text{if } j \neq q^* \end{cases}$$

where $num(i, j)$ represents how many times the user i asked query j and $0 < \alpha < 1$ weights queries different from q^*.

We define queries in function of the pages that they answer:

$$q_j = \sum_{k=1}^{s} b_{jk} r_k, \quad \text{where } b_{jk} = \begin{cases} sim(j,k) & \text{if } j = q^* \\ \beta sim(j,k) & \text{if } j \neq q^* \end{cases}$$

where $sim(j, k)$ is the similarity of page k with query j and $0 < \beta < 1$ weights queries different from q^*.

Finally, pages are defined by the clicks done by users:

$$r_k = \sum_{i=1}^{m} c_{ki} u_i, \quad \text{where } c_{ki} = hit(k, i, \{q^*\}) + \gamma hit(k, i, Q - \{q*\})$$

where $hit(k, i, S)$ is the number of times that user i clicked in page k while doing a query in S and $0 < \gamma < 1$ weights the queries different from q^*. In matrix form we have

$$\vec{u} = A\vec{q}, \quad \vec{q} = B\vec{r}, \quad \vec{r} = C\vec{u}$$

which implies

$$\vec{u} = A\vec{q} = A(B\vec{r}) = A(B(C\vec{u})) = (ABC)\vec{u}$$

$$q = B\vec{r} = B(C\vec{u}) = B(C(A\vec{q})) = (BCA)\vec{q}$$
$$\vec{r} = C\vec{u} = C(A\vec{q}) = C(A(B\vec{r})) = (CAB)\vec{r}$$

This defines an iterative method to compute u, q, and r. With this we obtain a ranking for the η_k values.

The parameter T is crucial for the effectiveness of the algorithm. If T is too large, the matrices are large and the algorithm is slower. In addition we add many queries that are not related to $\vec{q}^*$. On the other hand, if T is small, the method does not work and we are wasting useful log information. In fact, T is inversely proportional to the query frequency.

5.2 Experimental Results

MASEL was proposed for an image search engine, where the precision of answers is low. To give some simple examples, we tried this idea for text, where the results were quite good [4]. Typically, the first two or three answers were a good complement to the search engine ranking. For example for the word $cars$ (in Spanish) we used $T = 24$ and $T = 48$ hours. For 24 hours the results were:

```
www.chileautos.cl/link.asp
www.chileautos.cl/personal.htm
www.autoscampos.cl/frmencabau.htm
rehue.csociales.uchile.cl/publicaciones/moebio/07/bar02.htm
```

The first three are relevant, while the last one is not. For 48 hours we had:

```
www.mapchile.cl/
fid.conicyt.cl/acreditacion/normas.htm
www.clancomunicaciones.cl/muni_vdm/consejo.htm
```

where the last two are not relevant, but the first is relevant without containing the word $cars$, because is a site of maps. This example shows the potential of MASEL to find semantic relations through two persons that did different questions. It also shows the sensitivity of the value T.

Figure 7 shows the precision for three Spanish words of different frequency of occurrence: $software$, $banco$ (bank), and $empresas$ (enterprises). The most popular word, software, has good results with smaller T. Bank, for $T = 72$, has an excellent precision, which decreases with larger T. This is because bank has many meanings and the main meaning refers to a commercial bank. However, with larger T, other meanings start to appear in the query log. Clearly, the value of T depends on the frequency of the word and if the word is polysemic or not. In this case, as in link analysis, we can have a *topic drifting* in the answers (this was the case for bank).

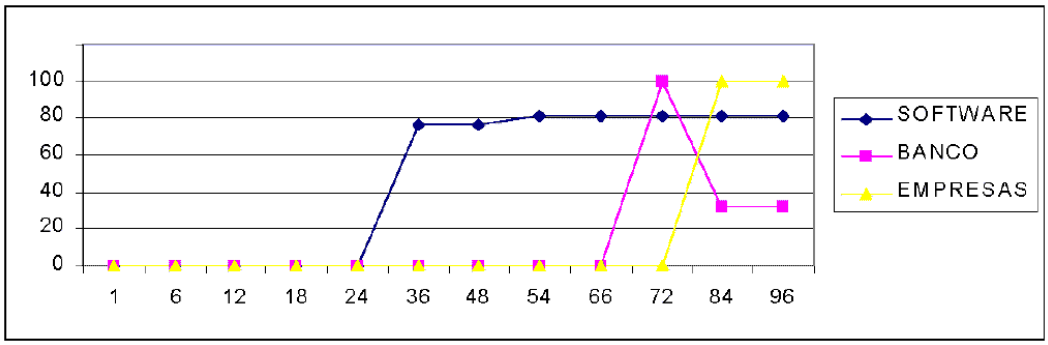

Figure 7: Precision of the answer versus T for three Spanish words.

If the word does not have the dominant meaning ir it is not frequent at all, this technique is not useful, as there is not enough data in a log with several days. The issue of topic drifting could be reduced by changing the parameters α, β, and γ.

6 Concluding Remarks

In our examples there are several lines for further research. They include better understanding of queries and their evolution in time; analysis of how the changes in the query distribution affects the performance of a given index layout; study the best combination of index layout based in query distribution, dynamic caching and compression; and to include Boolean and phrase queries in the index layout problem.

We are currently doing additional research in Web query mining to improve Web findability and information scent, as well as using queries to focus Web crawlers. For example, if we represent a given interest by a query vector Q using the vector model [1], a crawler can try to maximize the similarity of a retrieved vector page p with Q. This technique is used by focused crawlers or searching agents[3]. We can extend the idea by representing all past queries in a search engine as a vector Q_t, which is updated using a time based average, with the last queries q by using a moving average: $Q_{t+1} = \alpha Q_t + (1 - \alpha)q$, where α weights past versus current queries.

References

[1] Ricardo Baeza-Yates, and Berthier Ribeiro-Neto. *Modern Information Retrieval*, ACM Press/Addison-Wesley, England, 513 pages, 1999. URL: `http://sunsite.dcc.uchile.cl/irbook/`.

[2] Ricardo Baeza-Yates and Carlos Castillo. Relating web structure and user search behavior (extended poster). In *10th World Wide Web Conference*, Hong Kong, China, May 2001.

[3] Ricardo Baeza-Yates, Jose Miguel Piquer. Agents, Crawlers and Web Retrieval, Cooperative Information Agents (CIA) 2002, LNCS, Madrid, September 2002.

[4] Ricardo Baeza-Yates, and Felipe Saint-Jean. Query analysis in a search engine and its application to rank Web pages (in Spanish), BID 10, June 2003.

[5] Ricardo Baeza-Yates, and Felipe Saint-Jean. A Three Level Search Engine Index based in Query Log Distribution. SPIRE 2003, Manaus, October 2003.

[6] Ricardo Baeza-Yates, Barbara Poblete, Felipe Saint-Jean. The Evolution of the Chilean Web: 2001-2002 (in Spanish). Center for Web Research (www.cwr.cl), University of Chile, 2003.

[7] Doug Beeferman, and Adam Berger. Agglomerative clustering of a search engine query log, Proceedings on the 2000 Conference on Knowledge Discovery and Data Mining (Boston,MA), pages 407-416, Aug. 2000

[8] Soumen Chakrabarti, Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann Publishers, San Francisco, CA, USA, 2002.

[9] Brian Davison, David Deschenes, and David Lewanda. Finding Relevant Website Queries (poster), WWW12, Budapest, Hungary, May 20 - 24, 2003.

[10] Directhit: Home Page, www.directhit.com, 1997.

[11] Christoph Hlscher, and Gerhard Strube. Web Search Behavior of Internet Experts and Newbies, WWW9, Amsterdam, Netherlands, May 15 - 19, 2000.

[12] Jon Kleinberg. Authoritative sources in a hyperlinked environment. Proc. 9th Symposium on Discrete Algorithms, 1998.

[13] Mark Levene and Alex Poulovassilis, editors. Web Dynamics, Springer, to appear.

[14] Evangelos P. Markatos. On Caching Search Engine Query Results. In Proceedings of the 5th International Web Caching and Content Delivery Workshop, May 2000.

[15] Iko Pramudiono, Takahiko Shintani, Katsumi Takahashi, and Masaru Kitsuregawa. User Behavior Analysis of Location Aware Search Engine, Mobile Data Management, 139-145, 2002.

[16] Patricia Correia Saraiva, Edleno Silva de Moura, Nivio Ziviani, Wagner Meira, Rodrigo Fonseca, and Berthier Ribeiro-Neto. Rank-preserving two-level caching for scalable search engines, In Proceedings of the 24th annual international ACM Conference on Research and Development in Information Retrieval, New Orleans, USA, 51-58, September 2001.

[17] Peter Pirolli. Computational Models of Information Scent-Following in a Very Large Browsable Text Collection, In Human Factors in Computing Systems: Proceedings of the CHI '97 Conference. ACM Press, New York, 3-10, 1997.

[18] Andreas Schaale, Carsten Wulf-Mathies, and Sonke Lieberam-Schmidt. A new approach to relevancy in Internet searching - the "Vox Populi Algorithm", arXiv.org e-Print archive, August 2003.

[19] Craig Silverstein, Monika Henzinger, Hannes Marais, and Michael Moricz. Analysis of a Very Large AltaVista Query Log. *SIGIR Forum* 33(3), 6-12, 1999.

[20] Amanda Spink, Dietmar Wolfram, Bernard J. Jansen, and Tefko Saracevic. Searching the Web: the public and their queries. Journal of the American Society for Information Science and Technology, 52(3), 226-234, 2001.

[21] Amanda Spink, Bernard J. Jansen, Dietmar Wolfram, and Tefko Saracevic. From E-Sex to E-Commerce: Web Search Changes. *IEEE Computer* 35(3): 107-109, 2002.

[22] Amanda Spink, Seda Ozmutlu, Huseyin C. Ozmutlu, and Bernard J. Jansen. U.S. Versus European Web Searching Trends. SIGIR Forum 26(2), 2002.

[23] TodoCL: Home Page, www.todocl.cl, 2000.

[24] J-R. Wen, J-Y. Nie, and H-J. Zhang. Clustering user queries of a search engine. WWW10, Hong Kong, May 2001.

[25] Dietmar Wolfram. A Query-Level Examination of End User Searching Behaviour on the Excite Search Engine. Proceedings of the 28th Annual Conference Canadian Association for Information Science, 2000.

[26] Y. Xie, and D. O'Hallaron, Locality in Search Engine Queries and Its Implications for Caching, Infocom 2002.

[27] G-R. Xue, H-J. Zeng, Z. Chen, W-Y. Ma, and C-J. Lu. Log Mining to Improve the Performance of Site Search, 1st Int. Workshop for Enhanced Web Search (MEWS 2002), Singapore, Dec 2002, IEEE CS Press, 238-245.

[28] Dell Zhang, and Yisheng Dong. A Novel Web Usage Mining Approach For Search Engine. *Computer Networks*, 2003, to appear.

[29] George Zipf. Selective Studies and the Principle of Relative Frequency in Language, Harvard University Press, Cambridge, MA, 1932.