Inquiro.CL: a New Search Engine in Chile

Marcelo Mendoza* marcelo.mendoza@uv.cl
Hipólito Guerrero hipolito.guerrero@uv.cl
Julio Farias julio.farias@uv.cl
Department of Computer Science
University of Valparaíso
Avda. Errázuriz 1834, Valparaíso, Chile

ABSTRACT

In this paper we present a new online search engine developed in Chile: Inquiro.CL. This new search engine lets users search the Latin-American web (specifically, ten Spanish-speaking countries) by specifying their search domain (.cl, .com.ar, .com.mx, and the rest of Latin America). The structure is based on a distributed architecture that manages two document collections. The first is a cache of pages / sites that provides responses at low computational cost. The second is a general collection of documents that is visited when the user extends the search to the second results page or beyond. The index has been built using a multi-tier strategy, such that titles, URLs, headers, and complete site contents that make up the collection are cataloged in different indexes. The search engine uses a ranking function that combines various relevance measurements, among them user preferences, page rank, and text. Currently Inquiro's main collection reaches more than 1,500,000 pages and approximately 35,000 sites. Experimental results show that the search engine is precise and compares favorably with similar search engines, reaching an average of over 60% precision in the top 5 rankings.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation, Information Filtering, Search Process*

General Terms

Algorithms, Experimentation

Keywords

Search Engines, Latin American Web

1. INTRODUCTION

Due to the explosive growth of the Internet, today it is unthinkable to search the web without the help of a search engine. In Latin America we have two types of search engines to make this navigation easier: general purpose search engines, like Google or Yahoo! Search, which completely index the web and permit queries in specific domains, and vertical search engines that index websites according to a specific subject or specific domains. Examples of the latter type are TodoCL, a Chilean engine that searches only sites in the .CL domain, or the former TodoBR which did the same on the Brazilian domain. Since 2005, a team from the Universidad de Valparaíso Department of Computer Science has been developing a new search engine designed for the web in Spanish. The ultimate goal is to provide a better tool for searching by domain and which also helps evaluate new ranking algorithms. With the initial phase of the search engine complete, the site is now available at *http://www.inquiro.cl*.

1.1 Contributions

In this work, our experience in implementing a search engine for the Latin American web is presented. In this process we have evaluated three ranking functions and have designed distributed storage strategies, conducting experiments that allow us to visualize the advantages and disadvantages of each technique considered. Thus, the result is a new search engine that compares favorably in precision to other similar systems. We hope our discussion of results and the conclusions we have drawn will be useful to other groups working on developing search engine technologies, as well as academics and researchers in the area.

1.2 Outline

The rest of the paper is organized as follows: in Section 2 we present the related works, in Section 3 we describe the design of the search engine, in Section 4 we show experimental results of the proposed engine's performance compared to other similar models. Finally, in Section 5 we present our conclusions and future work.

2. RELATED WORK

There are two types of search engines that lend to domain-specific web searches: general purpose search engines (universal) and vertical search engines. Universal search engines index the web entirely and allow for domain-specific searches. This is the case for Google and Yahoo! Search, among other commercial search engines. In particular, Google [10], began at Stanford University as a spin-off, is implemented on a centralized architecture for indexing and crawling, which allows it to index a significant portion of the web in just a few months. It operates on a ranking algorithm based on PageRank [5], among other document relevance measures. PageRank is a measure of site visibility according to connectedness of the source in terms of hyperlinks. In par-

*Dr. Mendoza is now at Yahoo! Research, Santiago, Chile

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

ticular, to provide Latin American domain-specific searches, the Google system manages local collections in data centers, where those queries are directed.

Yahoo! Search [9] is a search engine that belongs to Yahoo! Inc. and is currently another important search engine on the web. Originally, Yahoo! Search began as a web directory organized hierarchically that meant searching for sites associated to specific topics. Later, Yahoo! Search incorporated a search engine that operated on a collection of pages indexed by crawling, and on the collection of sites recommended in its directory, thus combining both types of results. Yahoo! Search, like Google, allows users to conduct Latin American domain-specific searches, using local collections for each domain.

At the Latin American level, the site-specific search engines that standout are TodoCL and Mexico Global. TodoCL [11] is a search engine developed as part of family of search engines that include the former TodoBR, for sites in Brazil. TodoCL offers its users a search engine and a web directory affiliated with the OpenDirectory Project initiative. Because it searches the Chilean web, TodoCL can be considered a vertical search engine by location. TodoCL works by locally storing a copy of most Chilean websites, aiding the search for information and allowing for quick access. TodoCL's main objective is to provide precise information quickly, for which it consults an index of 600,000 indexed pages. This web search has gone through several improvements to date, in terms of interface, crawler and ranking. Today it is considered the most important search engine in Chile.

Mexico Global [12] is a search engine powered by a web directory of .com.mx domain-specific sites. The directory is organized hierarchically with 16 categories in the first level. The search engine recommends categories related to the query and categorized websites. This hybrid strategy for recommendations has also been adopted by the Colombian search engine Conexcol.com and the Cuban Cubaweb.cl. Conexcol [13] powers its searches by combining the results provided by the search engine with those from a local directory of 14 principal categories. Cubaweb [15] is primarily focused on facilitating searches for tourism, so it is considered a vertical search engine. Cubaweb allows users access to a directory of 12 main categories and provides access to sites related to specific places in Cuba. The search engine only recommends sites found in its local directory.

3. SEARCH ENGINE DESIGN

At the start of 2005, a group of academics and students from our Department of Computer Science decided to focus on designing a search engine dedicated to the Latin American internet written in Spanish (that is, excluding sites written in Portuguese) by specifying the domains on which the sites are hosted.

A first version was developed using a centralized architecture and considering one server to hold the collection and handle the search engine. The ranking function used was text-based, using the model $\texttt{Tf} - \texttt{Idf}$ [4] and the PageRank measure was incorporated into this function. The results in the first precision tests were disappointing, achieving average precision between 25% and 30%, according to evaluations based on expert criteria. Starting at the end of 2006, a second version was developed whose main objective was to make the following improvements: 1) Restructure the search engine by introducing distributed process architecture, 2) Improve the search engine's precision by including other measures of relevance. With regards to the architecture of the search engine, the downloading and collection indexing were separated from the search engine itself. The first of these improvements was called Inquiro, which lends its name to the search engine, and the second improvement was called Sator. In this new versions of the search engine, Inquiro will only store a cache of web pages / sites which will be pulled from a main repository of data managed by Sator. Thus, we hope to improve the search engine response times, given that it will refer to a collection smaller than the complete collection.

3.1 Search engine architecture

The project is composed of four subsystems operating cooperatively together. Two of these are characterized by the way they work and carry out their processes: the first operates offline while the second runs online. We understand the offline process as everything occurring before interaction with the user (before the query is processed), while online implies direct interaction with the user (processing the query and preparing the ranking). The other two subsystems are characterized by their support of the site distribution processes in both collections.

The offline subsystem is responsible for generating the information necessary for the entire system to work, performing tasks like crawling and indexing. The online subsystem is responsible for the ranking and subsequent display of results. The mediating subsystems are responsible for distributing the documents between the two main modules (Inquiro and Sator) and maintaining the primary collections and cache.

Figure 1 shows the structure of the system expanded to show subsystems and a description of the information flow. As shown in the figure, the indexing module that stores the documents collected in the primary index performs the task of crawling by using an additional module called WIRE. WIRE [8] is a set of freeware tools developed by the Center for Web Research [14], which allow a crawler to be used and to extract statistics from the crawling process and the documents indexed. On the other hand, the Inquiro and Sator modules communicate through their intermediary modules, whose objective is to maintain both document collections (cache and primary index).

Next the internal structure of each subsystem is described:

3.1.1 Query Engine

Composed of 4 modules: 1) Vectorized user query module: responsible for vectorizing the user's query. This module processes the query, vectorizing and storing it in the cache. 2) Relevant document search module: responsible for retrieving documents relevant to the query from the cache. 3) Results ranking module: responsible for ordering the documents retrieved from the collections (ranking process). For this process, the documents are ranked through a combination of relevant measures, among them: PageRank, user preferences registered in the logs (clicks) and the similarity of the document to the query using $\texttt{Tf} - \texttt{Idf}$. 4) Document list module: responsible for the pagination of the references retrieved from the cache and the primary index. By default each page result shows ten sites relevant to the query.

3.1.2 Inquiro mediator

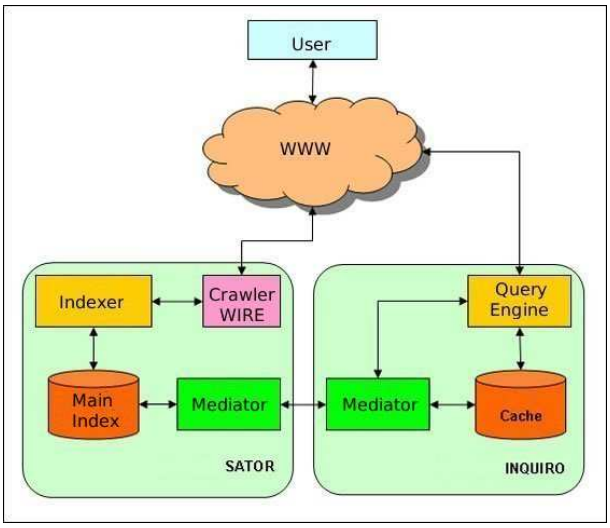

Figure 1: Distributed indexer-query engine architecture

Composed of two modules: 1) Query administrator module: responsible for verifying that the quantity of results obtained from the cache is enough to be displayed on the first page. If not, it requests more documents from the primary index to complete the top-ten ranking and the rest of the page results. 2) Document distribution module: responsible for maintaining the database cache. Only the ten most popular documents belonging to the top-k most frequent queries (a value of the k parameter is defined in the experiments section).

3.1.3 Sator mediator

Composed of two modules: 1) Query administrator module: responsible for asking the primary index for pages / sites relevant to a query. 2) Document distribution module: sends the cache the top ten most popular documents for the top-k most frequent queries.

3.1.4 Indexer

Composed of two modules: 1) Indexer module: responsible for maintaining the primary index updated and incorporating new sites / pages. 2) Collection download module: responsible for communicating with WIRE. In this module the parameters are configured to for the crawling process, such as the quantity of process iterations, the ccTLD where the crawling process will occur (crawling by domain), the number of pages / sites to download, the list of seed pages / sites, the maximum number of sites to visit, among other parameters in the process.

3.2 Logic Design

In this section we explain the design criteria that were used to define three sensitive functions of our search engine: the distribution of data between subsystems, index maintenance, and the design of the ranking function.

3.2.1 Data distribution

The proposed search engine administers two independent collections, through which searches are conducted. The first is a cache collection, maintained in the Inquiro subsystem, which is rapidly accessible given that it handles a smaller index. The second is a general collection stored on the Sator subsystem, which has a slower access than the cache collection because of its larger index size. This later collection is remotely stored, so it requires a remote invocation of methods. The collections are stored in mixed format, that is, the index is stored in text files and the rest of the data that describe the documents and queries, like clicks, PageRank and URLs among others, are stored in related databases. Two modules are responsible for coordinating the distribution of data between the two collections: a mediator stored in the Sator subsystem and another in the Inquiro subsystem. The Inquiro mediator asks the Sator mediator for the documents that will allow it to complete the list of results that has been created considering only sites / pages from the cache collection. The Inquiro mediator prioritizes these requests to Sator, giving greater priority to those queries that have not been able to show the first page of results to the user (top ten results). In addition to this function that supports the query processing, the Inquiro mediator is responsible for incrementing the click counter associated with each selected document, whether it is in the local collection (cache) or otherwise notifying the change to the Sator mediator to update the general collection. Finally, the Inquiro mediator is responsible for determining the documents that should form part of the cache, registering the top-k most popular queries and retrieving the top ten ranked documents, storing them in the cache if they are in the general collection. The value of the k parameter is determined in the experiments section.

The Sator mediator is responsible for responding to the Inquiro mediator's requirements. Also, along with the Inquiro mediator it is responsible for distributing the sites / pages between the two collections, leaving the most visited in the cache and keeping the rest of the collection in the primary index it administers. Upon receiving a request from Inquiro to complete a list of pages / sites recommended for a query, it runs a search in the primary index and produces a list of recommendations which is then sent to Inquiro for ranking. It also receives the notifications from Inquiro to add to the document click counter, modifying its value in Sator's related database.

Both mediators should communicate every so often to recalculate the top-k most frequent queries, to recalculate the top ten of each of those, and to update both indexes. The updating period is determined in the experiments section.

3.2.2 Indexer

Both the cache index and the primary index are structured in multi-tiers. The first tier corresponds to an index that contains the titles of the sites / pages indexed by Sator. A second tier contains the URLs. The third tier contains the site / page header (the first five lines). Finally, a fourth tier contains the content (complete text) of the indexed sites / pages. The implementation of multi-tier indexes allows us to build relevant document lists, weighting differently similarities in titles, URLs, headers and content in the ranking. This also facilitates and accelerates the document search because the set of ranked documents has a maximum of 1,000, discarding the rest of the rankable documents. To determine the set of 1,000 rankable documents we first look in

the title tier, then at the URLs, then in the header tier, and finally in the content tier. This allows us to implement a relaxed search strategy, since we can discard searches in later tiers if the first 1,000 documents have been retrieved in the present tier. The index is built using an incremental algorithm as a base, which allows new documents to be indexed without discarding the previous index. Because two different collections are maintained and given the redistribution of sites / pages conducted by the mediators that periodically remove documents from one index to put into another, we have added a reverse index to the primary index. That is, iddoc - idword, so as to facilitate the elimination of documents, avoiding the need to scan the entire index.

3.2.3 *Ranking*

The ranking was implemented using a combination of various types of relevance measures. First, we consider a scheme $\texttt{Tf} - \texttt{Idf}$ to estimate the relevance of a document to a text-based query. The similarity function used is the Cosine similarity, which is applied to each tier of text in the index (title, URL, header, body text). With that, we get 4 measures of relevance. Another measure of relevance considered is PageRank [5]. The value of PageRank for each document has been recovered using the WIRE module attached to our search engine. Given that the PageRank values are very low, we use a watered-down logarithmic version of its value given by $\texttt{PR}_{\texttt{soft}}(\texttt{u}) = \frac{1 + \log_{10}(\texttt{PR}(\texttt{u}))}{1 + \log_{10}(\texttt{PR}_{\texttt{MAX}}(\texttt{u}))}$, which is standardized for the maximum PageRank value in the collection. This allows $\texttt{PR}_{\texttt{soft}}(\texttt{u}) \in [0, 1]$. Finally, the last measure of relevance considered in the ranking function relates to user preferences. Claypool *et al.* [3] have shown that user preferences can be interpreted as implicit measures of relevance. In the Inquiro ranking we will consider the unbiased version of user preferences with respect to the position of the documents in recommendation lists, according to the model proposed by Baeza-Yates *et al.* [2]. Following the model for bias reduction, user preferences adjusted according to their bias to the position are given by: $\texttt{AdjPop}(\texttt{u}, \texttt{q}) = \texttt{Pop}(\texttt{u}, \texttt{q}) \times \texttt{r}^{\texttt{b}}$, where $\texttt{Pop}(\texttt{u}, \texttt{q})$ corresponds to the fraction of clicks from u document over all the sessions of q; r corresponds to the position of u in the raking of q; and b is a parameter of the model. Experimental results approximate the value of $\widehat{b} = 1.44$.

To combine the six measures of relevance considered in the ranking function, we use a linear combination. Let $\texttt{Sim}_{\texttt{title}}(\texttt{u}, \texttt{q}), \texttt{Sim}_{\texttt{URL}}(\texttt{u}, \texttt{q}), \texttt{Sim}_{\texttt{header}}(\texttt{u}, \texttt{q})$ and $\texttt{Sim}_{\texttt{full}}(\texttt{u}, \texttt{q})$ be the Cosine similarities using the $\texttt{Tf} - \texttt{Idf}$ scheme for the query, title, URL, header, and document content respectively. Let $\texttt{PR}_{\texttt{soft}}(\texttt{u})$ and $\texttt{AdjPop}(\texttt{u}, \texttt{q})$ be the PageRank measurements and Popularity adjusted respectively. The score of document u with respect to query q is given by:

$$\begin{aligned} R(u,q) &= \alpha \times \texttt{Sim}_{\texttt{title}}(\texttt{u}, \texttt{q}) + \beta \times \texttt{Sim}_{\texttt{URL}}(\texttt{u}, \texttt{q}) \\ &+ \gamma \times \texttt{Sim}_{\texttt{header}}(\texttt{u}, \texttt{q}) + \delta \times \texttt{Sim}_{\texttt{full}}(\texttt{u}, \texttt{q}) \\ &+ \epsilon \times \texttt{PR}_{\texttt{soft}}(\texttt{u}) + \zeta \times \texttt{AdjPop}(\texttt{u}, \texttt{q}), \end{aligned}$$

where α, β, γ, δ, ϵ and ζ are the factors of the linear combination of measures, where $\alpha+\beta+\gamma+\delta+\epsilon+\zeta = 1$. The values of these factors are determined in the experiments section.

3.3 User Interfaces

Figure 2: Query engine user interface

Figure 3: Answer list user interface

This section describes the design of the human-machine interface for the subsystems that compose our search engine. Of the four subsystems that make up our search engine, only two of them interact with the user: the indexer, which is used by the system administrator, and the query engine, which is used by the end-user. To develop the user interface of the query engine, the design objective was to be as intuitive as possible and to offer great ease of use. This subsystem has two interfaces: the query input interface and the results display interface. The navigation through these interfaces is oriented towards a request-response system, facilitating the use of the search engine. In the input interface the user can restrict the domain to search to .cl, .com.ar, .com.mx or the rest of Latin America, as shown in Figure 2.

The results display interface shows ten sites per results page, showing in the upper right corner the text search box. Each recommendation shows the title, URL and a snippet built from the header (the same that is considered for ranking, with a maximum length of two lines). In the page header the number of results retrieved and the query processing time in milliseconds are shown. At the foot of the page is the pagination of results. The interface is shown in Figure 3.

With respect to the design of the indexer interface for system administration, it is important to consider that this user has greater knowledge about the internal processes of a search engine. Thus the interface can use terminology specific to the area and can describe more precisely the features that can be accessed from within. These interfaces were developed using interactive menu styles, which lessen the training time needed to learn how the system works. In this stage, 6 interfaces were developed: login, main menu, download collection, collection indexing, statistics, and lo-

Figure 4: Indexer user interface

gout. Figure 4 shows the download collection interface that interacts with WIRE and allows parameters for crawling to be specified, such as the location of the seed file and the maximum number of documents to retrieve.

4. IMPLEMENTATION

In its first version, the system was developed in Java using J2SE 1.5 over DBMS PostgreSql 8.2 that allowed information about queries and documents like PageRank to be stored. The new version uses a DBMS PostgreSql 8.3. Both Sator and Inquiro have been developed using Java J2SE 6, including RMI for remotely calling the mediators procedures. In particular, the Sator module was developed on Java Swing. Inquiro runs on Apache Tomcat 5.5, given that part our application works using Servlets. For the development we have used IDE NetBeans and PGAdmin III as GUI for administering the related databases.

We use the 0.14 version of WIRE. FreeBSD version 6.2 was used as operative system for Inquiro and Sator.

5. EXPERIMENTAL RESULTS

The Sator application was evaluated using standard software evaluation methodology, considering unitary testing, integration testing, and system testing, where test had positive results. In this section we will explain in detail the experimental results with respect to three basic functions of a search engine: crawling, indexing, and ranking. The results that will be shown affirm that the search engine presented compares favorably with its peers.

5.1 Crawling

Crawlers scan and collect pages / sites from the web from a set of seeds, from which, following the structure of web hyperlinks, they index and collect new sites / pages referenced from the seeds. This process repeats on the new pages, following in turn the forward links. Different hyperlink scanning strategies can be defined. Basically, the structure of web hyperlinks defines a graph, which can be run in breadth or depth. The site / page servers define usage policies that prohibit an individual crawler from making a large amount

cctld	Sites	Pages	Words	Crawling time [m]
.ar	7,483	148,692	21,515,800	6,205
.cl	11,569	400,007	48,781,035	991
.co	2,349	166,962	15,968,117	3,315
.cr	972	155,235	17,445,040	4,188
.ec	841	66,917	13,731,681	660
.mx	6,192	182,231	19,493,372	4,996
.pe	2,015	139,135	27,951,729	4,016
.py	842	50,794	11,045,728	1,643
.sv	557	100,551	11,892,702	2,137
.ve	2,030	173,824	20,887,288	4,877

Table 1: Crawling results by country code domain.

of requests and consuming the page resources. Therefore, in this sense crawling strategies must be respectful. In general, a good scheduler strategy is scanning in breadth. However, for domain indexing, Baeza-Yates *et al.* [6] have proven that a better strategy may be ordering the pages by PageRank value. We will use this strategy in our search engine.

The crawler implemented through WIRE lets a scheduler be defined according to the PageRank of the downloaded site / page. To be able to start, the crawler needs to define the domain to be indexed and a list of seeds from which the process can start. In our case, we have decided to collect ten domains: Argentina (.ar), Chile (.cl),Colombia (.co), Costa Rica (.cr), Ecuador (.ec), Mexico (.mx), Peru(.pe), Paraguay (.py), El Salvador (.sv) and Venezuela (.ve).

The crawling process was run from a Lanix Spine 5015MT server with a Pentium processor d940, with a 3.2Ghz Dual Core with 2 GB RAM and two HDD Sata of 250 GB each, over a network with limited bandwidth of 300 KBPS. Given the size constraints we had (500 GB for the main index), the crawling process will be conducted with band width restrictions and a maximum number of document downloads per domain. We will give certain priority to the.cl domain with 1,000,000 sites and a maximum of 50,000,000 document downloads, as opposed to a greater restriction of 100,000 sites and a download limit of 1,000,000 pages for the remaining nine domains. The collection language has been Spanish, and we have considered a list of 50 stopwords, ISO-8859-1 codification, and PageRank score of 100, so that only documents with a high PageRank score are downloaded (give nour size constraints).

Table 1 shows the results for the completed crawling process. Each row shows the number of downloaded sites, the number of downloaded documents, the number of words indexed, and the download time in minutes, per domain.

In total, the collection indexed by Inquiro considers 34,850 sites, 1,584,348 pages, and 208,721,492 words. The processes that took the longest to complete were the Argentine web (app. 5 days), the Mexican web (app. 4 days), and the Venezuelan web (a little less than 3 days). The text preprocessing was supported with the help of the Swish-e package [7], which contributed to the construction of the main index.

5.2 Ranking

Given that our ranking function results from a combination of six measures of relevance, we will assign weights to each of these measures so they can be prioritized. Since we do not have evidence that will allow us to weigh one crite-

rion significantly more than another, we will assign values to the relatively similar coefficients of the score function. We will evaluate three weight assignments, based on the six coefficients of the score function, which are shown in Table 2.

Functions (1) and (2) give similar weights to the six criteria, with the only difference being in function (2), the elimination of the clicks measure to consider the impact of this criterion. Similarly, function (3) does not consider the use of multi-tier text, instead considering only measures of relevance for complete text and PageRank. Function (3) was originally used in the first version of Inquiro.

To evaluate document ranking, we will used the methodology described in [1], which consists of a calculation of performance measures for the top-10 recommended results. Thus, we have randomly chosen 30 queries from the 1,000 most frequent queries made at TodoCL during the second semester of 2006 (extracted from the search log), with the goal of being able to compare our results with the ranking of that search engine. Considering the above, the evaluation will be restricted to the domain .CL, understanding that this result will be generalized and extended to the rest of the Latin American collections. Likewise, we will conduct the comparative precision experiment considering Google restricted to .CL.

To evaluate the impact of user preferences, and given that at the moment of evaluation our search engine logs were not available (since it was not yet online), we asked 100 first-year Computer Engineering students to make queries and select the documents they thought were most relevant. Each student made 10 queries and checked the first results page shown by the search engine. The clicks were registered and incorporated into the score function calculation (1).

For each of the 30 query samples, we have determined the top 10 recommendations using our method. Seventeen members of our laboratory have evaluated the document recommendations made by our search engine as well as Google and TodoCL, according to their relevance to the query. This has required the evaluation of 712 document pairs - queries (there are documents that are recommended by more than one search engine, for which there is overlap). For each category, the documents recommended by the three search engines have been mixed up, so that the expert evaluations are not biased by their position in the list. Each expert has evaluated 10 queries, approximately equivalent to the evaluation of relevance of 237 documents. To achieve this, the 17 evaluators have used an evaluation page that allows them to express their opinions of relevance on a scale of 0 - 4, from less to greater relevance. Then, for each document pair - query, we have calculated the average relevance from the pairs evaluators. Of the 712 total pairs evaluated, 689 received evaluations in agreement (that is, 0 or 1 in the case of no relevance, 3 or 4 in the case of relevance). The remaining 23 pairs were reevaluated by a group of 4 different experts (who did not participate in the first evaluation), reaching a consensus on all the remaining pairs.

The precision results by query have been averaged for each evaluator. Then, for each position, we have calculated the average of the 30 queries, yielding the average precision graph shown in Figure 5.

As shown in the average precision graph, Google earned first place, followed by TodoCL and then Inquiro (1), with comparable precisions and in the range $[0.6, 0.8]$, which makes

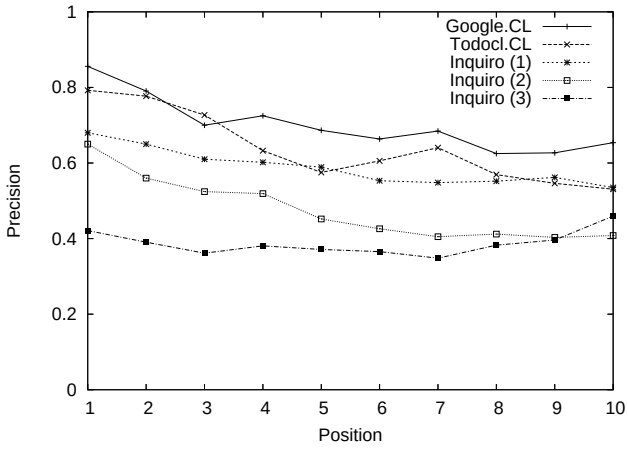

Figure 5: Average precision evaluation

our search engine favorably comparable with its peers. The graph also shows that the average precision does not drop significantly as the evaluated result ranks lower. The function with the lowest performance has been Inquiro (3), which can be attributed to not using clicks nor text in multi-tiers, as do Inquiro (2) and (1). Particularly, the inclusion of clicks has caused the performance of Inquiro (1) to surpass Inquiro (2) in precision, leaving it close to the curve achieved by TodoCL.

5.3 Data Distribution

Firsts, we will evaluate the size that the cache collection should be in relation to the quantity of most popular queries considered (top-k). To do this, we will measure the response time of the cache collection. This variable is determined by two factors: the time to search the collection and the time to display the results. To identify the impact of both factors on our measurements, we will make the same query 1,000 times, measuring the response time in two scenarios: 1) total system response time, 2) response time with a limit on the number of results displayed (in this case, top-10). This way, in the first scenario we are measuring the total time (search time plus display time) and in the second, we are only considering the search time. The query made was *Universidad Valparaiso Chile*, which has 3,441 resulting sites / pages in the collection. Table 3 shows the results achieved with this experiment.

The first column shows the number of queries considered in the cache, the second shows the number of sites indexed, the third lists the number of pages indexed, the fourth the average response time needed for Inquiro to show the first results page and the fifth the search time on the cache. As we can see, the average response time increases with respect to the cache size. This is because the costs of the search increase as the size of the collection increases. However, we can also observe that until a size of 5,000 queries, the time difference is not significant, which changes from 7,500 and up. We will therefore choose a cache with 5,000 queries (app. 50,000 indexed documents). Once the size of the cache collection is determined, we can evaluate the impact of the data distribution strategy used in Inquiro's design. For that, we

Function	Title (α)	URL (β)	Header (γ)	Full-text (δ)	Clicks (ϵ)	PageRank (ζ)
(1)	0.20	0.20	0.18	0.14	0.14	0.14
(2)	0.24	0.24	0.20	0.16	-	0.16
(3)	-	-	-	0.5	-	0.5

Table 2: Score functions evaluated in our experiments

will measure the search engine response times in accessing the main index and accessing only the cache, simulating various users connecting to our search engine. To conduct these tests, we will use JMeter, a tool which allows us to simulate user connections through the creation of threads. The experiment considered 1,000 requests. The cache collection returns lower response times than accessing the main index. The cache collection reached 5,870 requests per minute with an average response time of 306 milliseconds, as opposed to accessing the main index, which reached 66 requests per minute with an average response time of 904 milliseconds.

We also must evaluate the mediators response times. Particularly, a critical factor is the response time of the Inquiro mediator in communicating to the Sator mediator that it needs more documents to complete a query's results list. After 1,000 requests to the Inquiro mediator, the average response time was 2.03 milliseconds. Similarly, the Sator mediator response time was evaluated. With 1,000 requests, the average time was 110.69 milliseconds. On average in this test, the Sator mediator had to process around 3.400 documents per request, with a complete search of the multi-tier index (without relaxed search and without top 1000 restriction).

Finally, we will evaluate how much time is needed to update both indexes. The updating process considers the calculation of the top-k most popular queries, the top-10 ranking for each (since the ranking function includes clicks and therefore can affect the ranking), and finally the updating of the main index and cache. To determine the updating period, we will run the following experiment: With the cache collection empty of indexed documents, and restricted to the .cl domain, for the 5,000 most popular queries on TodoCL, we will generate 1,800 clicks every 12 hours, according to the click distribution by rank (power law) [2]. We will consider clicks in this experiment to simulate the effect of users using our search engine and selecting the shown documents with a bias towards rank position. With this, we can measure the effect of the index updates on the ranking function. This is because by considering clicks in the relevancy calculation, there will be documents that stop being popular (they leave the top-10 list and are therefore eliminated from the cache and are inserted in the main index) and other documents that will join the top-10 ranking (being extracted from the main index and inserted into the cache).

We will consider the following as updating periods: 12 hours, 1 day, 3 days, 7 days, 15 days, and 30 days. In Table 4 we show the number of documents that join and leave the top-10 ranking, according to the updating periods to evaluate.

We can see in Table 4 that the cost of updating the indexes increases as the updating period is greater. However, as the cache fills up, the number of documents that leave / join the ranking are less. Therefore, as the longer observation periods are considered, the updating costs relative to the cost of the first days of operation will be less. That is, on

Queries	Sites	Pages	Tot. cost [ms]	Search cost [ms]
100	25	921	466	234
500	103	4,853	473	270
1,000	243	9,365	468	224
5,000	1,012	46,862	472	282
7,500	1,375	72,957	484	372
10,000	2,152	96,102	491	295

Table 3: Time response by cache size.

Clicks	Deleted docs	Inserted docs	Total
1,800 (12 hours)	0	1,145	1,145
3,600 (1 day)	0	1,873	1,873
10,800 (3 days)	1	2,868	2,869
25,200 (7 days)	8	3,860	3,866
54.000 (15 days)	64	4,552	4,617
108.000 (30 days)	216	5,051	5,267

Table 4: Index updating costs

average, between days 15 and 30, 43 documents are updated daily, as compared to the first days (12 hours or 1 day) where around 2,000 documents need to be updated. For this, in the first 15 days we use a shorter updating period, and as the engine gets more use, the updating period will decrease. In practice, during the first month we have used daily updates. After the first month we have used additional updates every fifteen days.

6. CONCLUSIONS

We have presented our experience in implementing a new search engine designed for the Latin American web and for domain restriction. The collection contains over 34,850 sites, 1,584,384 pages, and 208,712,492 words from the crawling and indexing done over one month. The index has been built using a multi-tier technique, indexing titles, URLs, headers and body text in different tiers. The ranking function used by Inquiro combines six measures of relevance, four of which are related to the text layers of index sites / pages, in addition to PageRank which lets us consider the structure of web hyperlinks and clicks. The average precision achieved by our ranking function compares favorably to the precisions achieved by Google and TodoCL. Finally we have evaluated our distribution strategy, determining that a cache considering 5,000 queries (app. 50,000 documents in the cache) is enough. We also determined that when running our search engine, we should update both indexes daily, increasing that period to every 15 days once we have surpassed the first month of use.

There are several ways to improve Inquiro. Possibly the most relevant of all is that we run experiments that will allow us to define a ranking function that achieves precision equal to or better than similar search engines. What we have learned in this work is that all the measures of relevance used are useful and allow us to improve our ranking

function. What we still do not know is how to combine those measures of relevance. In this work we have taken on this problem using a lineal combination of measures, but we cannot deny that other combination strategies might produce better results. This problem can be met by following a machine learning scheme, where the coefficients of the combination are learned.

We will also work to improve the search engine so that it can process Boolean operators and also be able to process phrasal queries. At this time, the index only processes simple words, for which the processing of two and three words is a substantial improvement to make.

Finally, we will extend our collection to the rest of the Latin American countries, including more domains than were able to be included in this first version, like the Spanish web, among others.

7. ACKNOWLEDGMENTS

We would like to thank four undergraduate students Jade Gonzalez, Pablo Rompentin, Claudio Romo and Walter Paredes, for working on an initial implementation of the Inquiro Search Engine. This work was funded under DIPUV project 52/07 from Universidad de Valparaiso, Chile.

8. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, ACM Press, New York, 1999.

[2] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Improving search engines by query clustering. *J. Am. Soc. Inf. Sci. Technol.*, 58(12):1793–1804, 2007.

[3] M. Claypool, D. Brown, P. Le, and M. Waseda. Inferring user interest. *IEEE Internet Computing*, 5(6):32–39, 2001.

[4] G. Salton and C. Buckley. Term-weighting approaches in automatic retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[5] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks*, 30(1-7): 107 – 117, 1998.

[6] R. Baeza-Yates and C. Castillo and M. Marín and A. Rodríguez. Crawling a country: better strategies than breadth-first for web page ordering. Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005, pp. 864-872.

[7] Swish-e, Simple web indexing system for humans. available on *http://www.swish-e.org*

[8] Center of Web research. Web Information REtrieval available on http://www.cwr.cl/projects/WIRE/

[9] Yahoo! Inc.. Yahoo Search! available on http://www.yahoo.com

[10] Google Inc.. Google available on http://www.google.com

[11] Akwan TIC.. TodoCL available on http://www.todocl.cl

[12] Victeck Inc.. MexicoGlobal available on http://www.mexicoglobal.com.mx

[13] Conexcol. Conexcol. available on http://www.conexcol.com

[14] Center of Web Research. http://www.cwr.cl

[15] CubaWeb. CubaWeb available on http://www.cubaweb.cu