

## Hybrid Architecture for Web Search Systems Based on Hierarchical Taxonomies\*

FIDEL CACHEDA, VICTOR CARNEIRO, CARMEN GUERRERO AND ANGEL VIÑA  
*Department of Information and Communications Technologies  
University of A Coruña  
A Coruña, 15071 Spain*

Search systems based on hierarchical taxonomies provide a specific type of search functionality that is not provided by conventional search engines. For instance, using a taxonomy, the user can look for documents related to just one of the categories of the taxonomy. This paper describes a hybrid data architecture that improves the performance of restricted searches for a few categories of a taxonomy. The proposed architecture is based on a hybrid data structure composed of an inverted file with multiple integrated signature files. A detailed analysis of superimposing codes on directed acyclic graphs proves that they adapt perfectly well to a search system based on a hierarchical ontology. Two variants are presented: the hybrid architecture with complete information and the hybrid architecture with partial information. The validity of this hybrid architecture was analyzed by developing and comparing it with a basic architecture. The performance of restricted queries is clearly improved, especially with the hybrid architecture with partial information. This variant outperformed by 50 % the basic architecture for all workload environments, with a slight reduction in performance for the lower levels of the graph.

**Keywords:** signature files, superimposing code, inverted file, hybrid data architecture, hierarchical taxonomies, performance evaluation

### 1. INTRODUCTION

The number of documents available on the World Wide Web is increasing daily. The estimated number of pages in 1999 was 800 million, generating 6 terabytes of text information (without taking into account other multimedia contents) [1], and nowadays, Google claims to index more than 4,000 million documents. Describing and organizing this vast number of pages is essential for realizing their full potential as an information resource.

The use of search systems based on hierarchical taxonomies is an important ingredient as is evident from the popularity of web directories, such as Yahoo! and the Open Directory Project. Also, there has been important research on automatic text categorization using different learning approaches [25], which could be used on corporate networks or customer and partner extranet sites to obtain document taxonomies.

From a general point of view, search systems based on a taxonomy allow the retrieval of information by using a query composed of keywords, by browsing the taxonomy, or by using a combination of both. The hierarchical structure of the taxonomy can be represented as a directed acyclic graph, since each category may have several parents.

---

Received April 13, 2004; revised August 11 & December 27, 2004; accepted January 20, 2005.

Communicated by Ming-Syan Chen.

\* This paper was partially supported by the CICYT of the Spanish government, under project TIC2001-0547.

Equally, one document may be catalogued in different categories, which provides great power and cataloguing flexibility. However, these systems have the added value of a search process combined with the navigation process called restricted search, which improves the quality of the results obtained. In a restricted search, the user firstly browses the directory hierarchy of categories until a convenient category is found and then submits his/her query. For this type of query, the results are restricted to relevant documents associated with a subgraph of the hierarchy, where the root is the category to which the user has navigated. Since it uses the selected category, the search engine is likely to return documents that are more suitable.

The focus of this work is the performance achieved in the search process, especially for restricted searches. The search process is based on an inverted file structure, which relates keywords to their associated documents, while the browsing process is based on an inverted file, which associates each category with its Web documents. On the other hand, restricted searches must combine results from both inverted files, leading to a worsening of the response time due to multiple, intensive merging operations.

This paper examines restricted searches in detail from the point of view of the data structures used and the performance obtained. Performance is improved by using a hybrid data architecture composed of an inverted file and dynamic signature files, where two variants are defined: the hybrid architecture with complete information and the one with partial information. The novelty of this work is its focus on restricted search performance, which has not been considered deeply before, and the main contribution of our work is our performance evaluation of the proposed hybrid architecture. We prove that the hybrid architecture is able to maintain the response time of standard searches and improve on the response time of restricted queries. Moreover, some new experiments were conducted and are discussed here to demonstrate the improvement achieved in the performance of restricted queries, with regard not only to the number of results but also the number of documents associated with the restricted category. Also, a detailed analysis of the performance of queries restricted to deeper categories of the hierarchy is included to prove that the variant with partial information outperforms the other variant, although the percentage of improvement is reduced for higher categories.

The paper is structured as follows. It starts with a discussion of related works. Section 3 provides a detailed description of a basic architecture, examining how the restricted queries are solved. Section 5 describes the proposed architecture and the implementation in detail. Section 6 presents a performance evaluation of the proposed architecture (considering both variants) versus that of the basic architecture. Finally, main conclusions are drawn in section 7.

## 2. RELATED WORKS

Our work is related to several previous ones. Firstly, the works in [12, 15, 19] and [23] reported different improvements in the search process obtained using categories, directories or ontologies. In [15], the authors described how the PageRank computation could be biased by using a small number of representative basic topics taken from the Open Directory Project, in order to allow the query to influence the link-based score. The results showed that the average precision for the rankings induced by the topic-sensitive

PageRank scores was substantially higher than that for the unbiased PageRank scores.

In [19], Liu *et al.* described how a web search could be personalized in order to incorporate the user's interests in to the query. They proposed a novel technique for mapping a user query to a set of categories that will represent the user's search intention and can be used to disambiguate the words in the user's query. Their experimental results show that this technique is both effective and efficient.

Scime and Kerschberg [23] presented a methodology, architecture and prototype for query construction that provides the user with a ranking choice based on the user's determination of importance. The user develops a taxonomy in which different components of the problem are represented as categories for solutions.

In [12], the authors suggested that a user who is searching for documents within a specific category and using a general purpose search engine might not find relevant documents. The authors presented an automated method for learning search-engine-specific query modifications that achieves very high precision for specific categories.

Our work is related to these previous works in the sense that all of them modify the traditional search process by using categories, ontologies or directories with the final objective of improving the final results obtained by users. However, our work focuses on the data structures used in this novel search process and the improvement achieved in performance. As we have noted, our architecture is based on hybrid data structures (using inverted files and signature files), and some previous articles have also explored the possibilities of these hybrid data structures, not in search systems based on hierarchical taxonomies, but in basic search systems. The works reported in [7, 9] and [11] explored the performance improvement that could be achieved by using a hybrid structure versus a standard inverted file, in different search systems.

Devros *et al.* [9] proposed a new data structure called S-Index (Signature-Index) based on the combination of inverted files and signature files based on superimposed codes. The proposed data structure can be adjusted to obtain a performance in the range of the inverted files and the signature files. The authors proposed the use of the S-Index in text retrieval, where the most common queries can be indexed using an inverted file (obtaining optimal performance), while the remaining terms can be stored using a signature file, which offers a better data compression.

Faloutsos *et al.* [11] analyzed the use of a hybrid system to improve response times, taking into account the biased distribution of the terms frequency. In their system, the most frequent terms replace the inverted list with a bit vector. All the bit vectors are stored together in a signature file. The results show that the hybrid system offers improved performance in static and dynamic environments, based on all main performance measures: space, response time and insertion time.

Cha *et al.* [7] described a mechanism for retrieving images based on their contents. Their index is based on a structure called HG-tree that can be combined with the signature files technique, thus leading to improved performance, especially when keywords are indexed.

### 3. BASIC ARCHITECTURE

This section describes the specific data structures for the components of a Web directory (as a general search system based on a hierarchical ontology) and presents a basic

architecture for determining fundamental design aspects for this type of information retrieval system.

Generally speaking, the number of descriptive articles related to IR on the Web is very small, especially with regard to Web directories. However, this architecture is based on similar models, such as those described in [3, 16-18].

A Web directory consists of three basic components that represent the information stored in it and the various interrelations (see Fig. 1). On one hand, the vocabulary stands for the keywords indexed in both the documents and categories. There is also a structure that represents the hierarchy of categories existing in the directory, typically composed of a directed acyclic graph. A small documents file is required with the basic information about each document (URL, title and description).

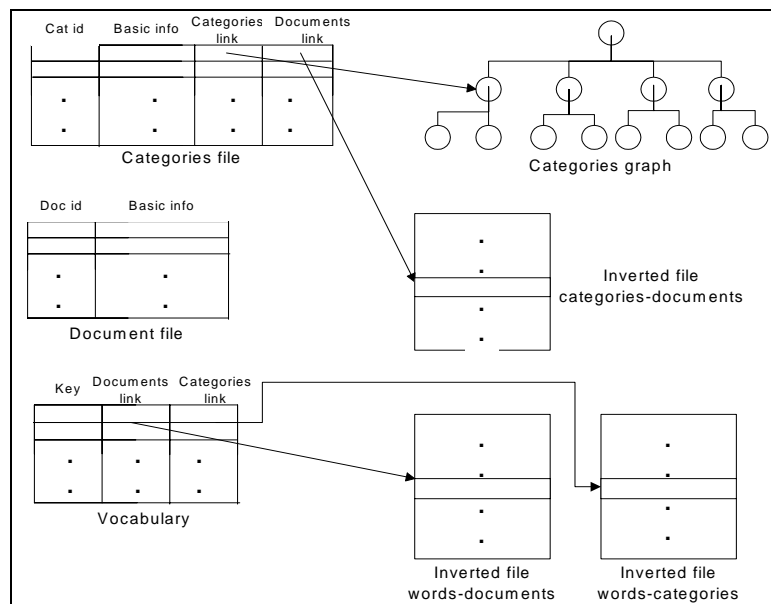


Fig. 1. Data structure of the basic architecture for a web directory.

With regard to the information that corresponds to the categories, each of them is identified by a single identifier. Thus, the basic information (name, brief description, etc.) is stored in a category file. Independently, a pointer-based structure based on the category identifiers represents the directed acyclic graph, which constitutes the ontology.

With regard to the keywords derived from the indexing process, the inverted file structure constitutes, at present, the indexing technique that achieves the best performance for huge data volumes [2]. Thus, keywords are stored in the vocabulary, using the best structure, such as ordered arrays, *B* trees, PAT structures, etc., together with the corresponding inverted lists [8, 13, 14].

Inverted lists may have different orders that directly influence the performance of the standard search process, and, in consequence, the restricted searches. An order ac-

ording to the document identifier facilitates the combination of several lists, while an order according to the criterion of document relevance makes simple searches trivial. There are also mixed alternatives, such as the one defined in [3] and used by Google.

Furthermore, for Web directories, the categories present keywords or descriptions, according to which they are indexed. As in the previous case, the inverted file structure constitutes the most efficient option, creating a common vocabulary. However, a set of occurrences or inverted lists independent of the documents inverted lists must be created for reasons of efficiency and system complexity.

As stated before, an inverted file structure relates keywords to documents. However, a structure that relates each category to associated documents is also necessary. This is intensely used during the browsing process. In this case, the inverted file structure also constitutes the most efficient option, so that the category file constitutes the vocabulary, while an inverted list of document identifiers is associated with each category. This list is preferably sorted according to the relevance or importance of each document.

### 3.1 Restricted Searches

Our basic architecture allows efficient processing of both standard keyword-based queries and directory-based browsing, using an inverted file as the basic data structure. In [27], the authors showed the use of inverted files in an indexing technique which achieves the best response time, requiring a 20% to 30% of the storage space of the original text, based on the use of several zipping techniques.

This indexing technique is widely used in the standard search process in different traditional commercial systems and on the Web [2]. Basically, this search process is divided into three steps. First, each term of the query is searched for in the vocabulary. Second, the occurrences associated with each word are retrieved. Finally, the different lists obtained are combined, taking into account logical or proximity operations.

The browsing process is analogous to the search one, although simplified, so the inverted file system achieves optimal performance by eliminating the last phase. During the browsing process, only the category is located (based on the identifier), and its list of associated documents is accessed without using any kind of combination operation. Optimal performance is achieved if the list has been previously ordered according to a relevance criterion.

The restricted search process in an area of the categories graph requires more elaborate access to that information. On the one hand, a standard search is carried out by accessing the inverted word file and combining the inverted documents lists in the usual way. Once the list of results has been calculated, the key step consists of determining which resulting documents belong to the specified graph area. Based on this basic data architecture, two approaches to filtering are defined.

The first one consists of obtaining the list of documents associated with the specified graph area. Later on, this inverted list is intersected with the list of results. Obtaining the list of documents included in a specific graph area is a difficult process and is divided into three steps:

- Initially, the graph is explored until every leaf node is reached, starting from the root node of the restriction area.

- A list of associated documents must be obtained for each node.
- Each and every one of the lists obtained in the previous step is combined.

The result obtained is a list of all the documents associated with the restricted graph area, ordered by an identifier. The next step is to intersect this list with the resulting list, in order to obtain only the resulting documents related to the restricted area.

The inverted files technique has a disadvantage that directly influences its performance: the combination (intersection or union) of two or more inverted lists causes the performance to deteriorate [21]. Assuming that the lists have been previously sorted according to the document identifier, the performance is inversely proportional to the number of combined lists and to the number of elements they have. Two important combination processes are required in this approach that will affect the performance: combining all the documents in the restricted graph area and combining the restricted documents list with the results list.

The second approach obtains the category list from the restriction area (an easier process than obtaining the document list) and checks the results list sequentially, that is, checks which documents are located at the nodes of the category list.

Obtaining the category list (ordered by an identifier) simply requires exploring part of the graph and storing the identifiers in an ordered list. Sequential exploration of the list of results does not require any previous order but may be especially hard if the list is extensive. In addition, an auxiliary structure indicating the categories associated with each document is required. This requires an inverted file relating each document to the categories it belongs to (the reciprocal of the index structure relating categories to documents, as previously defined).

Generally speaking, the first method is adapted adequately to those queries where the restricted graph area is reduced (deeper categories of the hierarchy), given that the number of involved categories (and, thus, of documents to be combined) is inferior. On the other hand, the second approach is efficient when the number of results obtained in the search is reduced (regardless of the amplitude of the restriction area), since both the sequential reading and the index access will be moderate.

However, neither of the two solutions efficiently deals with the searches that obtain a great number of results and that have been restricted to a wide graph area (the study reported in [4] shows that most searches are restricted to categories in the top-three levels of the hierarchy).

As a consequence, it is necessary to create a new data architecture that allows this type of query to be solved in an efficient way. This aspect is discussed in the next section.

#### 4. HYBRID ARCHITECTURE

From a descriptive point of view, the problem with restricted searches lies in obtaining an inverted list of results that then undergoes a filtering process based on the value of an attribute (associated categories). The disadvantage is that the attribute has a complex hierarchical structure, which makes filtering difficult.

An architecture of data structures is proposed here, based on the second alternative

for the basic architecture described above, for the purpose of solving this type of query in an efficient manner when the number of results to be examined is very high.

The proposed data architecture is based on the application of an inexact filter, which allows most of the results that are not associated with the restricted categories to be eliminated. Thus, exact filtering only examines the remaining documents, the number of which will have been considerably reduced.

The signature files technique adapts well to these features. Signature files are inexact when used in a filtering technique (as measured based on the false drop probability) based on sequential access to the data, which constitutes its main disadvantage that leads to poor performance. However, in this case, that is no inconvenience, since sequential filtering is only carried out using the search results, never using the whole set of documents in the collection.

In the proposed architecture, each document must have a signature, which represents each and every one of the categories it belongs to, directly or indirectly. Nevertheless, this does not imply the existence of a global signature file for every document. On the contrary, signature files are incorporated into inverted files, thus creating a hybrid scheme with inverted and signature files. Thus, in the proposed hybrid data structure, in order to guarantee efficient access, the signature file is embedded into the inverted one, so that each inverted list is a signature file as well. Therefore, the scheme for restricted searches is as follows:

- Firstly, the list of results is obtained through the standard search process, with no restrictions placed on their order (they are sorted usually according to a relevance order).
- Secondly, inexact filtering is carried out for the category to which the search is restricted, based on the signature file associated with the list of results.
- Thirdly, exact filtering of the remaining results is carried out, according to the second previously described.

In fact, the third step should be optional. When inexact filtering produces pure results, exact filtering can be omitted, thus reducing the response time for restricted queries. An important future work will be analysis of the precision and recall parameters based on the false drop probability of the inexact filter (avoiding the exact filter). However the present work focuses on the response time of the proposed system versus that of the basic architecture, with the precision and recall parameters kept equal for all the systems (by means of the exact filtering phase).

This constitutes the proposed hybrid architecture from a general perspective. However, certain aspects, such as the joint storing of both structures, the validity of signature files and the superimposing codes, have to be defined in order to represent the categories associated with a document. These aspects are dealt with in detail in the following sections.

#### **4.1 Signature Files and Superimposed Codes Applied to a Directed Acyclic Graph**

A key aspect of the development of the proposed architecture consists of using the signature file technique and the subsequent superimposed codes to represent the categories associated with each document. In essence, the signature file technique is based on

the superimposed codes originally developed by Mooers [20] and later expanded by Roberts [22]. In this technique, each document is divided into blocks that contain a constant number of different words. A signature is linked to each word, and the block signature is obtained by means of the bit to bit logic OR of its words signatures. A search is carried out, beginning with from the signature of the query term, just by making a logic AND between the term and documents signatures, while checking whether it coincides with the term's original signature.

The following design parameters are associated with this technique:

$D$ : the constant number of different words which are integrated into each block;

$b$ : the number of bits integrated into each signature (initially set to "0");

$w$ : the number of "1" bits in each signature.

The performance of superimposed coding when applied to information retrieval can be measured by using two different parameters: storage overhead and false drop probability (represented by  $F_d$ ). The first one concerns with the amount of space required to store the generated signature file. The latter represents the number of extra documents that must be reviewed in the search process but will not match the query; it is calculated as follows [10]:

$$F_d = \frac{\text{false drops}}{N - \text{actual drops}},$$

where,  $N$  is the total number of documents in the collection. *actual drops* means the number of documents retrieved by the query that actually match the query. *false drops* means the number of documents retrieved by the query that do not match the query.

It is obvious that these two parameters are contradictory: the smaller the space overhead is the higher will be the false drop probability and vice versa, in order to obtain a minimum false drop probability the space overhead will increase.

To obtain the minimum false drop probability, the weight of the signature must be much smaller than the number of bits in the signature. Stiasny proved that for a given signature size and a given number of words per block, the false drop probability is minimized if the number of "1"s is equal to the number of "0"s in the block signature [24]. By minimizing the false drop probability, we can obtain the optimal value for the weight of the signatures as described in the formula shown below. This optimal value leads to an equal number of "1"s and "0"s in the block signature, as proved by Stiasny; consequently, the false drop probability obtained for the optimal weight is

$$w = \frac{b \cdot \ln 2}{D} \Rightarrow F_d = 2^{-w}. \quad (1)$$

Based on this technique, several optimizations have been defined which were based on compression, horizontal or vertical storing techniques that are described in detail in [10].

Using this as a basis, we adapt the signature files to a Web directory. The signature file technique, when applied to a directed acyclic graph of categories, associates a dif-

ferent signature each category (each node in the graph), and each document generates its signature by superimposing the category signatures with which it is directly or indirectly associated [5].

Fig. 2 shows a simple example of a Web directory. The figure shows how document 2 belongs directly or indirectly to categories 5, 3, 2 and 1. Thus, for a restricted search in any of those categories, document 2 would be one of the possible results.

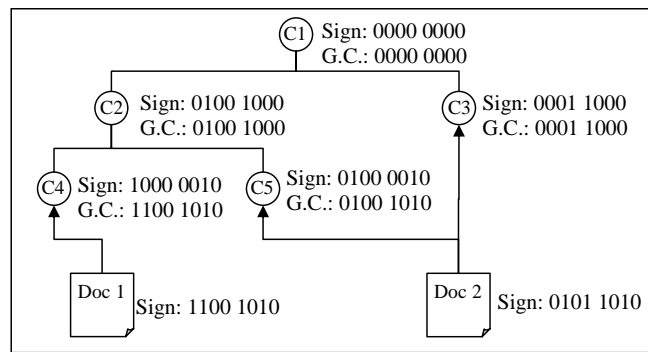


Fig. 2. Superimposed codes applied to a directed acyclic graph of categories.

Following the above example, each category has a unique signature, and the document signature is the result of superimposing the signatures of its categories. In the example, the signatures of categories 5, 3, 2 and 1 are, respectively (where  $b = 8$  bits and  $w = 2$  bits are assumed): 0100 0010, 0001 1000, 0100 1000 y 0000 0000 (the root category does not require a signature, since every document belongs to this category). Therefore, the document's signature is: 0101 1010.

A genetic code has been defined for each category (represented by G. C. in Fig. 2). It results from the superimposition of the signature of the category and all its ancestors. This genetic code contains much more information than a single signature and, so it is used in a search in the signature file, thus reducing the number of false drops. For instance, let us consider a search restricted to category 5 (with signature 0100 0010 and genetic code 0100 1010). Let us assume that standard search will retrieve documents 1 and 2. Firstly, let us consider that the signatures of documents 1 and 2 (1100 1010 and 0101 1010, respectively) are individually ANDed with the signature of the restricted category (0100 0010). In this case, both documents will be classified since the result of both AND operations will be the signature of the category. However, if the signatures of the documents are ANDed with the genetic code of the restricted category (0110 1010), then only document 2 will be classified for the restricted query, since the result of the AND operation is the genetic code of the category. This example proves the importance of genetic codes for reducing the number of false drops.

This basic example also illustrates the main differences between the proposed method and traditional signature superimposition. First of all, in traditional signature files, the value of  $D$  guarantees a constant number of superimposition operations. However, in this case, each document may belong to several categories, and one category may

present several parents. This implies that the number of superimposition operations in the document signatures is not constant.

The second difference lies in the values of  $D$ ,  $b$  and  $w$ . In traditional systems, it is assumed that  $D$  takes a high value, and that the signature size ( $b$ ) is on the order of several hundreds of bits, while  $w$  is only a few bits. In the proposed architecture, the value of  $D$  is directly related to the graph's depth, so it will never take a very high value, and we try to keep the value of  $w$  low, since this will affect the required storage space.

These differences imply that we cannot use the simplified formulas for the signature files described in Eq. (1) to estimate the false drop probability. Due to the importance of the false drop probability for queries restricted to all levels of the hierarchy, simulation techniques have been used to estimate these values. A prototype of the information retrieval system has been implemented in order to simulate and study the false drop probability for different types of queries.

In each simulation process, ten different trees were generated with approximately 2,000 categories 6 depth levels, using a collection of 50,000 documents. Each node had a signature consisting of 52 bits, with a weight of 6 bits (this is the optimal weight for  $b = 52$  and  $D = 6$ ). For each tree, we simulated a group of queries that were restricted to the categories of all the tree levels (called  $D_q$ ). Each group of queries consisted of 1,000 queries that retrieved about 1,000 documents. This configuration was the basis for the rest of the simulations described in this section.

Table 1 shows the results obtained when genetic codes were used to reduce the false drop probability as the query depth was increased. From this table, it is clear that the application of the genetic codes in inexact filtering reduced  $F_d$  as the query depth increased, since the amount of information included in the codes was greater. The false drop probability improved by one magnitude order per level dropped in the graph, whereas when only the signature information was used, the  $F_d$  value remained constant for all levels. This is a very important characteristic of this coding scheme, because it implies that the deeper the level at which the query is performed, the better performance achieved.

**Table 1. Effects of the genetic codes versus the signatures on the false drop probability.**

Query depth ( $D_q$ )	$F_d$ (signatures)	$F_d$ (genetic code)	% reduction
1 <sup>st</sup> level	0.0082108	0.0082108	0.0%
2 <sup>nd</sup> level	0.0082108	0.0007835	90.46%
3 <sup>rd</sup> level	0.0082108	0.0001573	98.08%
4 <sup>th</sup> level	0.0082108	0.0000395	99.52%
5 <sup>th</sup> level	0.0082108	0.0000040	99.95%

These first results demonstrated that the superimposed coding and the signature files could be effectively used to represent complex hierarchical data structures. However, to determine the parameters that could directly influence the performance of the signature files, a complete analysis was carried out. This analysis was extensively described in [5], and the parameters studied were: the categories and their structure, and the documents and their categories.

The simulations showed that the parameters that increase the number of superimpo-

sitions have a negative effect on  $F_d$ . These parameters are the percentage of documents associated with several categories, the percentage of categories with several parent nodes and the depth of the graph. These parameters tend to remain stable throughout the life of a Web directory, although if any important changes occur, the parameters of the superimposed codes should be reconsidered in order to avoid a decrease in the performance of the system. On the other hand, the false drop probability is independent of more dynamic parameters (such as the number of documents and categories in the search system). In conclusion, the superimposed codes used in the signature file technique can adapt perfectly well to a Web directory and its category graph environment, thus guaranteeing that the false drop probability will remain stable in the dynamic context of the search system.

#### 4.2 Hybrid Data Structure for Signature Files Storage

Now that the superimposed codes have been proved to be able to adapt correctly to a directed acyclic graph of categories, the next step is to decide where the signature files should be stored in order to guarantee efficient access to the signatures of the documents.

The first basic approach is to store each document in the superimposed code included in the documents file (see Fig. 1), thus creating a traditional signature file. This solution is quite simple, and the storage space increase is minimal; however, for every document, a disk access is required to obtain its superimposed code (which can be located in  $O(\log_2 N)$ ), which may affect the performance of the proposed architecture.

Ideally, the signature file associated with each query should be obtained dynamically without any new disk accesses and without repeating the merge operations done over the inverted query term lists.

For this purpose, a document identifier is proposed. This identifier is composed of the superimposed signature of every category with which the document is associated and of a local document identifier. Thus, those documents that belong to the same categories will have the same signature (obtained from the same superimposed codes) and will be distinguished by means of the local identifier.

This document identifier is the foundation for a hybrid structure consisting of an inverted file and signature files (see Fig. 3). From a global point of view the inverted file is still composed of inverted lists, but multiple signature files are also included in the inverted file.

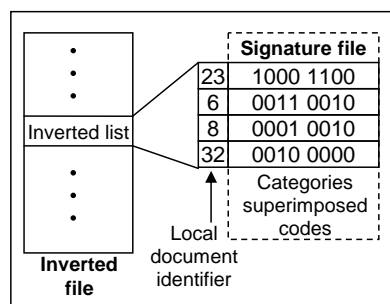


Fig. 3. Hybrid data structure of inverted file and signature files with composed document identifiers.

In this way, when the combination operations of the inverted lists are carried out (intersections, unions or any other operations) the associated signature file can be automatically obtained, dynamically generating the signature files for each query.

By means of this scheme, documents can be univocally identified, and all the combination operations carried out with the use of document identifiers can be simultaneously and automatically performed on the associated signatures. Thus, when the final list of document identifiers is obtained, the associated signature file will be indirectly available, regardless of the number and types of combination operations previously carried out and without any new disk accesses.

In the algorithm used to perform restricted queries, this novel hybrid data structure simplifies the second phase (inexact filtering for the restricted category) by using the signature file that is directly associated with the results, without repeatedly applying merge operations to the results and without performing new disk operations.

The main disadvantage of this method lies in the space required by the proposed identifier. The number of bytes required is a design parameter of the system that has to be fixed for all the documents in the collection. The signature assignment to the categories (and, consequently, to the documents) was described in the previous section, while the local document identifier follows consecutive numbering of the documents associated this signature. The increase in the storage space could be an important parameter affecting the performance of the system; thus, it will be examined in detail in section 6.

For the insertion of new documents in a directory, a new valid document identifier should be provided. This new document identifier can be obtained by first computing the superimposed code of all the categories associated with the document. Then, the first free local document identifier can be located either dynamically or by using a pool of free local document identifiers. When a document is updated, the document identifier must be updated only if the categories associated with the document are modified. In this case, the simplest solution is to remove the previous identifier and compute a new one.

## 5. IMPLEMENTATION OF THE HYBRID ARCHITECTURE

The main aspects of the hybrid data architecture consisting of an inverted file and signature files have already been defined. Two variants based on that architecture have been established. They are called the hybrid architecture with complete total information and the hybrid architecture with partial information. The sections describe in detail the two variants and the corresponding implementations.

The implementations that were carried out involved of developing a Web directory prototype based on a real environment, in our case, a Spanish Web directory called BIWE (available at <http://www.biwe.es/>). This prototype consists of a categories graph composed of approximately 1,000 categories distributed in 7 depth levels, where more than 51,000 Web documents have been classified. Each category has at least one parent node (except for the root category), and approximately 160 categories have more than one parent node. Each document classified in the system was associated with at least one category (not necessarily a leaf category), and nearly 14,000 documents are related to two or more categories.

The prototypes were developed based on a three-layer architecture, using Oracle as

the managing system for the database. Services were developed by means of Java Servlets and JSPs, which use several link APIs specifically designed for this environment. The development environment was an Ultra Enterprise 250 machine with a processor running at 300 MHz, 768 MB memory and 18 GB of storage space.

### 5.1 Hybrid Architecture with Complete Information

The hybrid architecture with complete information corresponds to direct application of the superimposed code technique to the categories graph. In this case, all of the categories have an associated signature (regardless of the depth level) therefore; they provide some information for their genetic code.

The first steps in the design and implementation of this hybrid architecture of inverted file and signature files were determining the parameters corresponding to the superimposing codes ( $D$ ,  $b$  and  $w$ ) and obtaining a suitable false drop probability for this system (estimated at less than 1%).

This kind of system does not establish a value for  $D$ ; it is determined by the structure of the categories graph. The value of  $b$  was set at 52 bits in order to obtain a document identifier value of 64 bits. To obtain the optimal value of  $w$ , simulation techniques were used. These techniques produced optimal values of  $w = 3$  or  $w = 4$ . The lower value was taken, since it accepts better an increase in the number of superimposing operations with the signatures.

The prototype was implemented based on these data and by generating and assigning the various codes to the categories and later calculating the identifier for each document. Thus, changes in the storage space requirements were identified.

Fig. 4 shows with a dotted line the new data structures or those structures whose sizes increased. The increase in size was due to the signatures associated with the categories as well as the new format of the document identifiers.

The incorporation of the signature into the information retrieval system only influenced the categories file (see Fig. 4 (a)), leading to an estimated increase in size of 10KB. On the other hand, the increase in the sizes of document identifiers caused a general increase in the sizes of several data structures. Compared to the initial size of 32 bits, the new identifiers required 64 bits each. This implied an increase of 200 KB in the size of the document file itself (see Fig. 4 (b)), while the indexes that related categories to documents (see Fig. 4 (c)) and keywords to documents (see Fig. 4 (d)) were duplicated. Obviously, this increase in size was more obvious in the inverted index, which associates keywords with documents, due to its great size (estimated, in this instance, at 12 MB, as opposed to the 6 MB size of the basic architecture), and due to the fact that it is used intensely in the standard search process, which may degrade performance since the number of reading times increases. This aspect will be determined at the performance evaluation.

Finally, based on the proposed architecture, a new structure such that, starting from a document, the categories to which it is directly associated can be obtained. This index (see Fig. 4 (e)), which is not present in the basic architecture, is smaller in size (around 275 KB), which is an acceptable system overload.

The maximum number of categories and documents supported by the system is important. Actually, each category must have a single signature, and each document a single identifier.

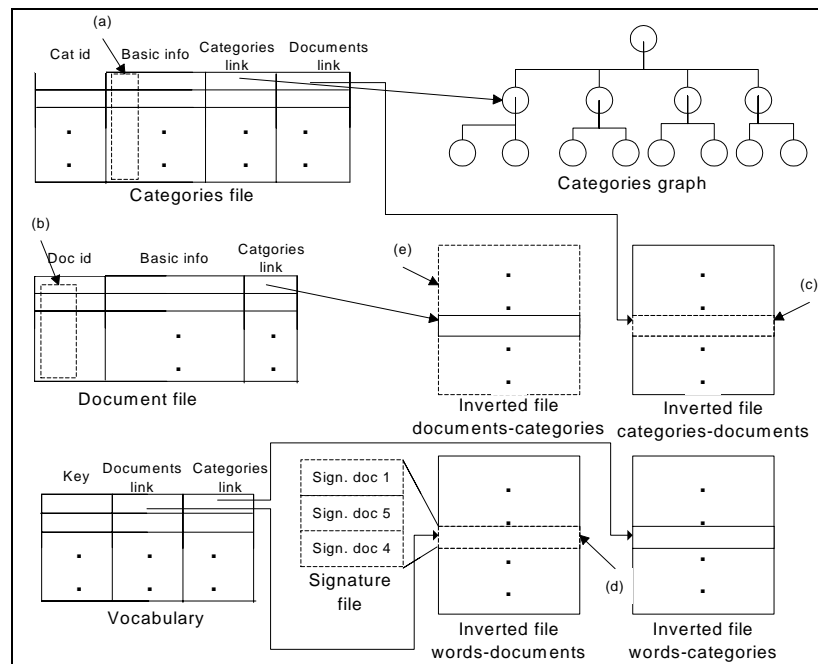


Fig. 4. Data structure of the hybrid architecture with inverted file and dynamic signature files.

In the implementation that was carried out, the number of categories was limited to 22,100 (assuming  $w = 3$ , since  $C(52, 3) = 22,100$ ), which perfectly supported the existing categories and possible future increases. Assuming  $w = 4$ , the limit was set above 270,000, which provided a sufficient margin for including all the categories in Yahoo!, estimated at 150,000 [17].

With regard to the number of documents, and considering the structure of the identifier which gives 12 bits to the local document identifier (4,096 documents for each signature), and nearing the number of genetic codes to the category signatures one, the system supported more than 90 million Web documents, which is two magnitude orders bigger than the estimated number of documents in Yahoo!

## 5.2 Hybrid Architecture with Partial Information

This section describes a variant of the proposed hybrid architecture which aims to reduce the sizes of the signatures and document identifiers, therefore reducing the required storage space.

According to Eq. (1), the false drop probability can be reduced by increasing the value of  $b$  or by decreasing the value of  $D$ . In traditional signature file applications,  $D$  takes a fixed value established by design. However, in the proposed architecture, it is possible to reduce the number of superimposing operations by applying the technique of superimposed codes only at certain levels of the graph (for example, the top-three levels), while allowing the rest of the nodes to inherit the signatures and genetic codes of the upper levels.

Therefore, the value of  $D$  is reduced, which allows values smaller than  $b$  to obtain similar values to those of  $F_d$  at the top levels, thus reducing the required storage space. This variation implies an increase of  $F_d$  at the lower levels, which should always be monitored with the goal of avoiding deterioration of the system's performance.

However, this disadvantage is lessened by two factors. A statistical analysis of restricted queries that was carried out showed that 80% of the queries were restricted to categories in the top-three levels [4]. In addition, the typical values of the false drop probability for the lower levels were three to four magnitude orders smaller.

In the implementation of this variant,  $w$  was estimated by taking a value of  $b = 32$  bits and fixing the maximum depth for signature assignment at three levels. The optimal value of  $F_d$  was found to be  $w = 3$  bits, with a slightly higher value of false success probability than in the previous case. Thus, a 46 bit composed document identifier was generated, with 14 bits for the local document identifier, and 32 bits for the categories signature.

Fig. 5 shows the obtained false drop probabilities for the variants with complete information and partial information. In this figure, it is clear that for the variant with partial information,  $F_d$  increases in the lower parts of the graph (levels 4, 5 and 6). But overall, this variant obtains homogenous values for false drop probability, regardless of the query depth (between 0.021 and 0.012 for all levels).

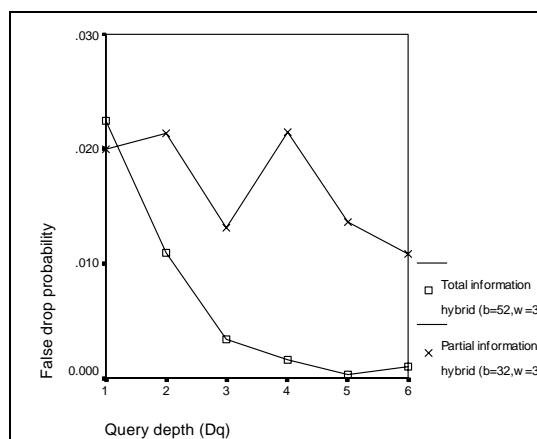


Fig. 5. Estimated false drop probabilities for the hybrid architecture with complete information and with partial information.

As for the increase in the required storage space, it is significantly smaller. Increases occur in the same structures that were described in the previous section, with an approximate reduction of 50% in the required storage space. It should be noted that the index of keywords and documents requires 65% less storage space, a decrease of 8.25 MB.

Furthermore, the hybrid variant with partial information is particularly flexible with regard to the number of categories it supports. The fixed parameters of  $b$  and  $w$  establish

a maximum of 4,960 different signatures ( $C(32, 3) = 4,960$ ) for the categories in the top-three levels and an unlimited number for the lower levels. This is a significant feature, because it allows the lower levels of the taxonomy to grow and to link without restrictions, which is not true for the previous architecture. With regard to the number of documents, the size of the local identifier was estimated in order to have a similar limitation to the previous case (a maximum of approximately 80 million documents).

## 6. PERFORMANCE EVALUATION

This section contrasts the response times obtained with the two variants, as opposed to a basic architecture based on the structure defined in Fig. 1.

The evaluation was designed to determine whether there is some kind of penalty in the standard search service with the hybrid model, as well as to check the performance improvement (if any) obtained with restricted queries.

The methodology used to evaluate the performance was adapted to the environment of information retrieval systems on the Web, which support variable load intensity through time. Five load situations were used to measure the response times: void, low, medium, high and saturation. This ensured a more realistic evaluation. In addition, several of the parameters described in [26] are indirectly evaluated.

The various load points were determined by finding the saturation points of the prototypes that were developed. The results were 5 searches per minute for low load, 12 searches for medium load, 19 searches for high load and 23 searches for saturation, with corresponding numbers of accesses to categories and documents.

Load generation was performed using a simulation tool especially designed for that purpose. This tool sends requests (searches, category accesses and document visits) to the search system, thus simulating the behavior of Internet users. Simultaneously, queries are sent to the search engine, and the response times are measured.

The queries used in the tests were real queries obtained from our previous query log analysis work described in [4]. The analysis of the response times obtained was based on an ANOVA (Analysis Of Variance) test, in which two factors (the number of results and the type of architecture) for standard searches and three factors (the number of results, the number of documents associated with a category and the type of architecture) for restricted searches were considered. The ANOVA test is a statistical method used to determine whether a significant relation exists between variables. In our case, the final objective was to determine if the average response times for the basic and hybrid architectures were different, taking into account the other parameters (the number of results and number of documents associated with a category).

In our previous work described in [6], a reduced performance evaluation of our hybrid architecture was performed, and this section presents some new and significant contributions: we compare the response times obtained with our hybrid architecture for restricted and standard queries. Some new experiments were also performed to analyze the improvement obtained with restricted queries, in terms of the number of results and the number of documents associated with the restricted category. A detailed analysis of the performance in the lower levels of the hierarchy is also included.

## 6.1 Standard Searches

Firstly, the response times of the various architectures for normal searches was analyzed. An ANOVA test was carried out to determine the influence of each architecture (basic, hybrid with complete information and hybrid with partial information). Two factors were considered: the number of results per query and the type of architecture. Obviously, the number of results is a very influential parameter, so our goal was to determine if the type of architecture influenced the response time for standard searches (due to a decrease in the performance for the hybrid architectures).

The tests were carried out under the same five load situations. The ANOVA test showed that the type of architecture did not influence the response time for a standard search, subject to a rather high probability level (approximately 80%) in all of the workload environments.

## 6.2 Restricted Searches

An evaluation of restricted searches was carried out under the same five workload environments, where queries were restricted to categories in the top-three levels of each prototype. Queries were retrieved from 0 to 2,000 documents and were restricted to categories with 50 to 20,000 associated documents.

As in the previous case, the analysis of the response times obtained was based on an ANOVA test, in which three factors were considered: the number of results, the number of documents associated with a category and the type of architecture. The first two are very influential parameters, so our objective was to determine if the type of architecture influenced the response time for restricted searches.

Firstly, response times were obtained for the different architectures, taking into account the number of results retrieved by a standard query. In this case, very similar results were obtained under void, low and medium loads, as shown in Figs. 6, 7, and 8, respectively. As can be seen in the three figures, the hybrid architectures clearly achieved about 50% better performance than the basic architecture in terms of the response time. This improvement was achieved when the number of results obtained in an ordinary search was high (over 500 results), since the performance of the basic architecture deteriorated dramatically in this case. In addition, the ANOVA test considered the type of system as an influent parameter with 99.9% of probabilities. The main difference under three workload environments was the overall increase in the response times when the system load increased.

To confirm the similarities between the hybrid architectures, the ANOVA test was repeated for the variant with complete information and the variant with partial information under void, low and medium loads. The results confirm that the response time for restricted queries is independent of the type of architecture.

On the other hand, under a high load, the behavior of the hybrid architectures varied considerably, as can be seen in Fig. 9. The performance of the hybrid architecture with complete information worsened significantly, approaching that of the basic architecture. An ANOVA test was carried out only with the results of the hybrid architectures. While in the previous cases, both behave similarly, this test revealed differences between the

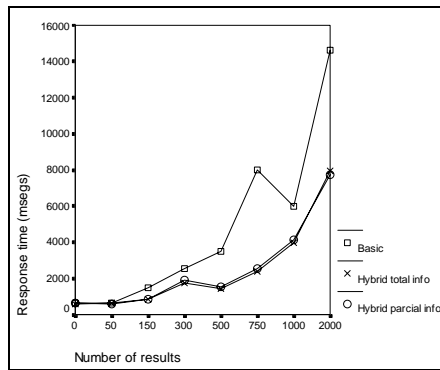


Fig. 6. Estimated response times according to the number of results (void load).

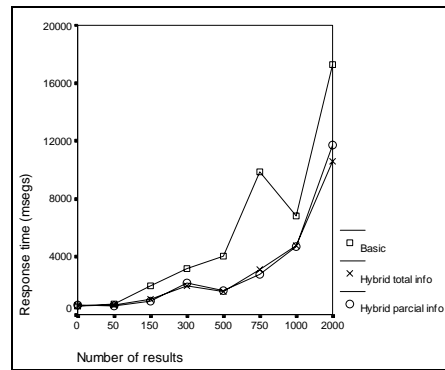


Fig. 7. Estimated response times according to the number of results (low load).

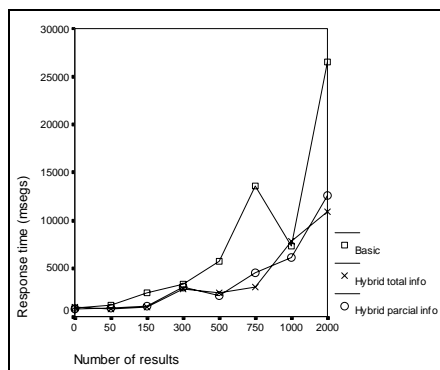


Fig. 8. Estimated response times according to the number of results (medium load).

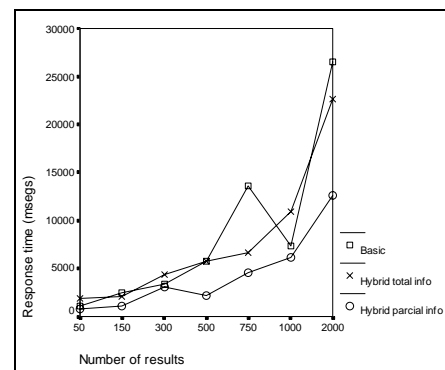


Fig. 9. Estimated response times according to the number of results (high load).

hybrid architectures: in the high load situation, the hybrid architecture with partial information outperformed the complete information variant (which still performed better than the basic architecture).

The experiments conducted under saturation did not produce conclusive results, due to the fact that the response times yielded by the system were much degraded and varied dramatically according to various uncontrolled parameters.

In the second part of the analysis, the response times obtained by the different architectures were evaluated by taking into account the number of documents associated with the restricted category. The results obtained were very similar to the previous ones and they are shown in Figs. 10, 11, 12, and 13.

In a void, low and medium workload environment (Figs. 10, 11, and 12, respectively) the hybrid architectures achieved improved response times compared to the basic architecture, especially for categories with more associated documents. In fact, for the top-level categories of the directory with more associated documents (approximately 20,000) the hybrid architectures had 50% lower response times compared to the basic architecture.

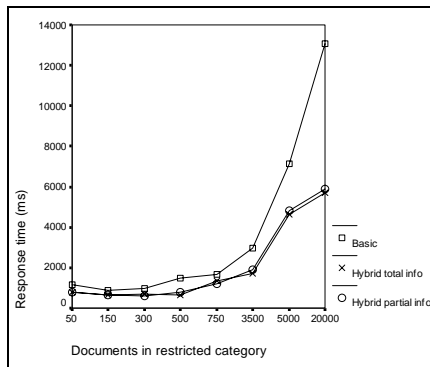


Fig. 10. Estimated response times according to the number of documents associated with the restricted category (void load).

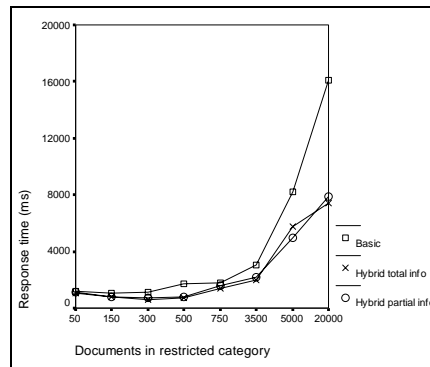


Fig. 11. Estimated response times according to the number of documents associated with the restricted category (low load).

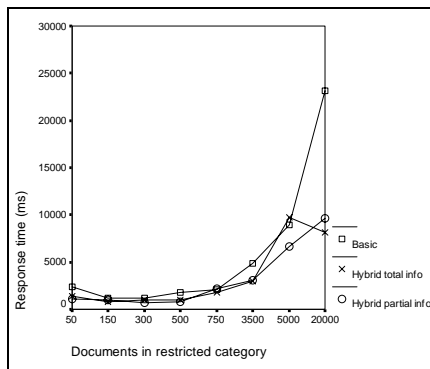


Fig. 12. Estimated response times according to the number of documents associated with the restricted category (medium load).

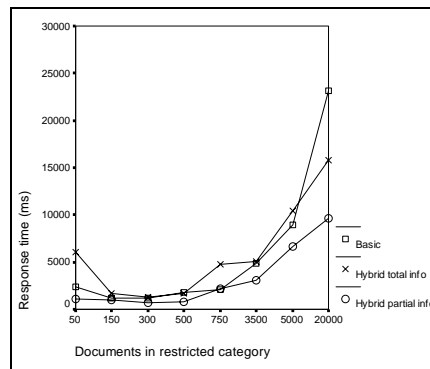


Fig. 13. Estimated response times according to the number of documents associated with the restricted category (high load).

It is worth to mentioning that the undulations present in the response times, as shown in Figs. 6, 7, 8, and 9 for the basic model and in Fig. 12 for the complete information model, were due to variations introduced by some minor factors not considered in this ANOVA model, such as the number of query terms or the query distribution, which had some effects, but did not impact the main results (in all cases, the correlation factor was higher than 0.85 for the ANOVA tests).

As in the previous case, the analysis conducted under a high workload (Fig. 13) showed that the performance of the hybrid architecture with complete information deteriorated greatly, leading to performance similar that of the basic architecture, while the hybrid architecture with partial information is able to achieve 50% improvement over the baseline.

The worst performance of the complete information architecture due to the exces-

sive size of the index of keywords and documents. A high load situation implies that a large number of searches reach the index, so the disk undergoes more reading operations.

In the proposed architectures, the inexact filtering application results in a reduction of the response times. However, in the complete information variant, the time required by the search process minimizes the positive effect of inexact filtering. Meanwhile, with the hybrid architecture with partial information, the search time is shorter (since it requires fewer accesses to the disk), adding to the benefits of inexact filtering and resulting in an improvement in the response time of up to 50% compared to the basic architecture.

The hybrid architecture with partial information achieved better performance; however, as was described previously, since no signatures are assigned to the lower graph levels, it might be penalizing the searches restricted to these levels.

Therefore, a new set of tests was conducted to analyze the response times of queries restricted up to the fifth level of the hierarchy. The ANOVA test examined the effects of three parameters: the number of results, the depth of the query and the type of architecture. The initial set of queries limited to queries retrieving more than 500 results to make more patent the differences among the architectures.

Tables 2 and 3 provide a summary of the results obtained in the void, low, medium and high workload environments. As in the previous tests, the experiments conducted in a saturated environment did not produce any conclusive results, because the response times varied severely according to various uncontrolled parameters.

**Table 2. Average response times for the basic architecture, hybrid architecture with complete information and hybrid architecture with partial information for restricted queries to the first 5 levels of the graph (void and low workload environments).**

Query depth	Void load			Low load		
	Basic	Total	Partial	Basic	Total	Partial
1	23527.06	11625.93	11396.47	27343.53	13875.73	13345.73
2	2259	1945.47	1765.27	3356.27	2168.73	2338.13
3	2197.8	1103.87	1017.27	2853.73	1304.07	1185.26
4	3093.67	933.8	870.8	3568.27	2730	1052.33
5	730.2	630.13	640.33	1146.07	740.13	707.8

**Table 3. Average response times for the basic architecture, hybrid architecture with complete information and hybrid architecture with partial information for restricted queries to the first 5 levels of the graph (medium and high workload environments).**

Query depth	Medium load			High load		
	Basic	Total	Partial	Basic	Total	Partial
1	36856.27	17327.2	19963.13	61243.4	35490.27	29565.8
2	3846.6	3434.8	2393.93	7241.73	5028.67	4995.33
3	3857.67	1696.47	1647.33	4386.4	8460.33	2491.33
4	5266.93	1459.47	1215.67	6621.67	3535.47	2512.93
5	1193.2	1117.53	1774.2	1817.67	2064.93	1836.73

The response times obtained for the top-three leveled match those obtained in the previous experiments: the performance of the variant with complete information deteriorated under a high workload, while the variant with partial information achieved 50% improvement in the response times compared to the basic architecture.

For queries restricted to the lower levels of the categories graph, the performance of the hybrid architectures seemed to remain stable. The deterioration of the performance of the hybrid architecture with complete information is still patent in the high workload environment, although the difference with the variant with partial information was slightly reduced.

Meanwhile, the analysis of the performance of the hybrid architecture with partial information for the lower levels of the graph (where the false drop probability was artificially increased) showed that there was a slight increase in the response times. In some cases (level 5 in the void and medium workload environments), the hybrid architecture with complete information outperforms the partial information variant, although the difference in the response times was not very significant. But in the high workload environment, the variant with partial information still outperformed the variant with complete information, although the percentage of improvement was reduced for the upper levels of the graph.

These results confirm that the increase of the false drop probability in the lower levels for the hybrid architecture with partial information did not impact the response times severely. As explained previously for the partial information variant, the fact that the false drop probability ratios were homogeneous for all levels enabled it to outperform the hybrid architecture with complete information (and the basic architecture) for the top-three levels of the categories graph. Meanwhile, the response times for the lower levels slightly reduce the performance improvement achieved over the basic and the hybrid architecture with total information.

## 7. CONCLUSIONS

This paper has analyzed in detail restricted searches in search systems based on hierarchical taxonomies (typically a Web directory) from the point of view of the data structures used and the performance achieved. A new hybrid data architecture has been proposed, which is composed of an inverted file and multiple signature files and is specially designed to improve the performance of restricted searches.

A detailed and exhaustive analysis of the use of superimposed codes in directed acyclic graphs has been provided. The results show how the hierarchical information can be easily applied to superimposed codes (through so-called genetic codes) to greatly reduce the false drop probability of restricted queries as the query depth increases (up to a magnitude order per level). Also, the signature generation techniques based on similarities with the parent node and/or siblings can reduce  $F_d$  value, especially for top level queries.

The parameters of the search system that could negatively affect the performance of the signature files (the percentage of documents and categories with several parents and the graph depth) tend to remain stable through out the life of the search system. Meanwhile, the more dynamic parameters (such as the number of categories and documents),

do not affect the false drop probability of the system. All these results confirm that the superimposed codes adapt perfectly well to the directed acyclic graph of categories of a search system based on a hierarchical ontology.

The proposed document identifier allows for the introduction of signature files into the system. This generates a hybrid data structure, with an inverted file and multiple integrated signature files, that will dynamically produce a signature file associated with the results of each query (without new disk accesses or new merge operations).

Two variants of the architecture have been defined: the hybrid architecture with complete information and the hybrid architecture with partial information. The former involves the direct application of the hybrid architecture described to the whole categories graph. The latter focuses on the top levels of the graph and considerably reduces the storage space required.

A detailed performance evaluation of restricted queries has been carried out using the two variants of the proposed architecture and a basic architecture, while taking into account different workload environments for IR systems.

The results show that the variant with complete information outperformed by 50% the basic architecture in a void, low load or medium load environment, while in a high workload situation, its response times were degraded, being equivalent to the baseline. On the other hand, the variant with partial information outperformed by 50% the basic architecture in all of the workload situations.

A more exhaustive study was conducted to analyze the performance of this variant in the lower levels of the graph. The results demonstrated that the variant with partial information still outperformed the baseline and the variant with complete information, although the percentage improvement was reduced for the upper levels.

The design and implementations of the hybrid architectures proved to be sufficiently flexible with regard to the number of documents and categories that the system can support. In fact, the variant with partial information only limits the number of categories for the top levels of the graph.

As in previous works presented in [7, 9] and [11], the use of a hybrid data structure has been shown here to improve the performance of the system. Although a direct comparison with the previous works is not possible due to the intrinsic differences in the search systems (image retrieval systems, text retrieval systems and search systems based on hierarchical taxonomies, in our case), in all of these cases, clear benefits are obtained by using the hybrid models.

Future research will focus on deeper analysis of the use of superimposed codes in directed acyclic graphs. In this work, some signature generation techniques have been outlined, although some other techniques should be considered with the goal of reducing the false drop probability. Also, in future works, exact filtering in restricted queries will be removed to see if this can lead to additional reduction in the response time, although its effect on the precision and recall parameters of restricted searches should also be carefully examined.

## REFERENCES

1. M. Agosti and M. Melucci, "Information retrieval on the web," in M. Agosti, F. Cre-

- stani, and G. Pasi, (eds.), *Lectures on Information Retrieval: 3rd European Summer School (ESSIR)*, LNCS 1980, 2001, pp. 242-285.
2. R. Baeza-Yates and B. Ribeiro-Neto, "Searching the web," *Modern Information Retrieval*, Addison Wesley, 1999, pp. 367-395.
  3. S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *Proceedings of 7th International World Wide Web Conference, Computer Networks and ISDN Systems*, Vol. 30, 1998, pp. 107-117.
  4. F. Cacheda and A. Viña, "Experiences retrieving information in the world wide web," in *Proceedings of 6th IEEE Symposium on Computers and Communications*, 2001, pp. 72-79.
  5. F. Cacheda and A. Viña, "Superimposing codes representing hierarchical information in web directories," in *Proceedings of 3rd International Workshop on Web Information and Data Management*, 2001, pp. 54-60.
  6. F. Cacheda and A. Viña, "Optimization of restricted searches in web directories using hybrid data structures," in *Proceedings of 25th European Conference on Information Retrieval Research*, LNCS 2633, 2003, pp. 436-451.
  7. G. H. Cha and C. W. Chung, "A new indexing scheme for content-based image retrieval," *Multimedia Tools and Applications*, Vol. 6, 1998, pp. 263-288.
  8. D. Cutting and J. Pedersen, "Optimizations for dynamic inverted index maintenance," in *Proceedings of 13th International Conference on Research and Development in Information Retrieval*, 1990, pp. 405-411.
  9. D. Dervos, P. Linardis, and Y. Manolopoulos, "S-index: a hybrid structure for text retrieval," in *Proceedings of 5th Conference on Advanced Databases and Information Systems*, 1997, pp. 204-209.
  10. C. Faloutsos and S. Christodoulakis, "Description and performance analysis of signature file methods," *ACM Transactions on Information Systems*, Vol. 5, 1987, pp. 237-257.
  11. C. Faloutsos and H. V. Jagadish, "Hybrid index organizations for text databases," in *Proceedings of 3rd International Conference on Extending Database Technology*, 1992, pp. 310-327.
  12. E. Glover, G. Flake, S. Lawrence, W. P. Birmingham, A. Kruger, C. L. Giles, and D. Pennock, "Improving category specific web search by learning query modifications," in *Proceedings of the IEEE Symposium on Applications on the Internet*, 2001, pp. 23-31.
  13. G. Gonnet, "Unstructured data bases or very efficient text searching," in *Proceedings of the ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, Vol. 2, 1983, pp. 117-124.
  14. D. Harmon, E. Fox, R. Baeza-Yates, and W. Lee, "Inverted files," in W. B. Frakes and R. Baeza-Yates, (eds.), *Information Retrieval: Data Structures and Algorithms*, Prentice-Hall, 1992, pp. 28-43.
  15. T. H. Haveliwala, "Topic-sensitive PageRank," in *Proceedings of 11th International World Wide Web Conference*, 2002, pp. 517-526.
  16. G. Jacobson, B. Krishnamurthy, D. Srivastava, and D. Suciú, "Focusing search in hierarchical structures with directory sets," in *Proceedings of 7th International Conference on Information and Knowledge Management*, 1998, pp. 1-9.
  17. Y. Labrou and T. Finin, "Yahoo! as an ontology – Using yahoo! Categories to de-

- scribe documents,” in *Proceedings of 8th International Conference on Information and Knowledge Management*, 1999, pp. 180-187.
18. W. Lam, M. Ruiz, and P. Srinivasan, “Automatic text categorization and its application to text retrieval,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, 1999, pp. 865-879.
  19. F. Liu, C. Yu, and W. Y. Meng, “Personalized web search by mapping user queries to categories,” in *Proceedings of 11th International Conference on Information and Knowledge Management*, 2002, pp. 558-565.
  20. C. Mooers, “Application of random codes to the gathering of statistical information,” Bulletin 31, Zator Co., Cambridge, Mass. Based on M.S. thesis, MIT, 1948.
  21. R. Baeza-Yates and B. Ribeiro-Neto, “Indexing and searching,” *Modern Information Retrieval*, ACM Press, Addison Wesley, 1999, pp. 191-228.
  22. C. S. Roberts, “Partial-match retrieval via the method of superimposed codes,” in *Proceedings of the IEEE*, Vol. 67, 1979, pp. 1624-1642.
  23. A. Scime and L. Kerschberg, “WebSifter: an ontology-based personalizable search agent for the web,” in *Proceedings of the International Conference on Digital Libraries: Research and Practice*, 2000, pp. 203-210.
  24. S. Stiasny, “Mathematical analysis of various superimposed coding methods,” *American Documentation*, Vol. 11, 1960, pp. 155-169.
  25. Y. Yang and X. Liu, “A re-examination of text categorization methods,” in *Proceedings of 22nd ACM International Conference on Research and Development in Information Retrieval*, 1999, pp. 42-49.
  26. J. Zobel, A. Moffat, and K. Ramamohanarao, “Guidelines for presentation and comparison of indexing techniques,” *ACM SIGMOD Record*, Vol. 25, 1996, pp. 10-15.
  27. J. Zobel, A. Moffat, and K. Ramamohanarao, “Inverted files versus signature files for text indexing,” *ACM Transactions on Database Systems*, Vol. 23, 1998, pp. 453-490.



**Fidel Cacheda** received his Ph.D. and B.S. degrees in Computer Science from the University of A Coruña, A Coruña, Spain, in 2002 and 1996, respectively. He has been an Assistant Professor in the Department of Information and Communication Technologies, the University of A Coruña, Spain, since 1998. From 1997 to 1998, he was an assistant in the Department of Computer Science of Alfonso X El Sabio University, Madrid, Spain. He has been involved in several research projects related to Web information retrieval and multimedia real time systems. His research interests include Web information retrieval and distributed architectures in information retrieval. He has published several papers in international journals and has participated in multiple international conferences.



**Victor Carneiro** received his Ph.D. and B.S. degrees in Computer Science from the University of A Coruña, A Coruña, Spain, in 1998 and 1993, respectively. He has been an Associate Professor in the Department of Information and Communication Technologies, the University of A Coruña, Spain, since 1995. He has participated in many research projects and in research on network management, distributed systems and information retrieval over the Internet. Currently, he is working on programmable networks oriented to efficient information retrieval over the Net.



**Carmen Guerrero** received the Telecommunication Engineering degree in 1994 from the Technical University of Madrid (UPM), Spain, and the Ph.D. in Computer Science in 1998 from the Universidade da Coruña (UDC), Spain. She has been an Assistant (1994-2000) and Assistant Professor (2000-2003) at Universidade da Coruña. She has been an Assistant Professor since 2003 at the Universidad Carlos III de Madrid (UC3M), teaching computer networks courses. She has been involved in several national and international research projects related to information retrieval, access networks, network management and advanced network and multimedia real time systems. Some of the recent research projects in which she has been involved are: MUSE: Multiservice Access Everywhere (IST 2004-507295); E-NEXT: Network of Excellence in Emerging Networking Technologies (IST-2004-506869). Professor Guerrero has published more than 10 papers in the field of advanced communications in journals and conference proceedings.



**Angel Viña** received his Ph.D. degree in Telecommunications Engineering from the University Polytechnic of Madrid. He is a Professor in the Department of Information and Communications Technologies of the University of A Coruña. He has been the head researcher in several projects related to: Web information retrieval, network management, real-time support for multimedia interactive systems and architectures for telemetric systems on the Internet.