An Ontology-based Index to Retrieve Documents with Geographic Information [⋆]

Miguel R. Luaces, Jose R. Paramá, Oscar Pedreira, and Diego Seco

Database Laboratory, University of A Coruña
Campus de Elviña, 15071 A Coruña, Spain
{luaces, parama, opedreira, dseco}@udc.es

Abstract. Both *Geographic Information Systems* and *Information Retrieval* have been very active research fields in the last decades. Lately, a new research field called *Geographic Information Retrieval* has appeared from the intersection of these two fields. The main goal of this field is to define index structures and techniques to efficiently store and retrieve documents using both the text and the geographic references contained within the text.
We present in this paper a new index structure that combines an inverted index, a spatial index, and an ontology-based structure. This structure improves the query capabilities of other proposals. In addition, we describe the architecture of a system for geographic information retrieval that uses this new index structure. This architecture defines a workflow for the extraction of the geographic references in the document.

1 Introduction

Although the research field of Information Retrieval [1] has been active for the last decades, the growing importance of Internet and the World Wide Web have made it one of the most important research fields nowadays. Many different index structures, compression techniques and retrieval algorithms have been proposed in the last few years. More importantly, these proposals have been widely used in the implementation of document databases, digital libraries, and web search engines.

Another field that has received much attention during the last years is the field of Geographic Information Systems [2]. Recent improvements in hardware have made the implementation of this type of systems affordable for many organizations. Furthermore, a cooperative effort has been undertaken by two international organizations (ISO [3] and the Open Geospatial Consortium [4]) to

[⋆] This work has been partially supported by "Ministerio de Educación y Ciencia" (PGE y FEDER) ref. TIN2006-16071-C03-03, by "Xunta de Galicia" ref. PGIDIT05SIN10502PR and ref. 2006/4, by "Ministerio de Educación y Ciencia" ref. AP-2006-03214 (FPU Program) for Oscar Pedreira, and by "Dirección Xeral de Ordenación e Calidade do Sistema Universitario de Galicia, da Consellería de Educación e Ordenación Universitaria-Xunta de Galicia" for Diego Seco.

define standards and specifications for interoperable systems. This effort is making possible that many public organizations are working on the construction of spatial data infrastructures [5] that will enable them to share their geographic information.

During the last decades these two research fields have advanced independently. However, many of the documents stored in digital libraries and document databases include geographic references within their texts. For example, news documents reference the place where the event happened and often the place where the document has been written. Furthermore, the information in a spatial data infrastructure often includes documents with geographic information such as construction licences or urban planning information. Finally, geographic references can also be attached to web pages by using information from the text, the location of the web server, and many other information elements.

Even though it is very common that textual and geographic information occur together in information systems, the geographic references of documents are rarely used in information retrieval systems. Few index structures or retrieval algorithms take into account the spatial nature of geographic references embedded within documents. Pure textual techniques focus only on the language aspects of the documents and pure spatial techniques focus only on the geographic aspects of the documents. None of them are suitable for a combined approach to information retrieval because they completely neglect the other type of information. As a result, there is a lack of system architectures, index structures and query languages that combine both types of information.

Some proposals have appeared recently [6–8] that define new index structures that take into account both the textual and the geographic aspects of a document. However, there are some specific particularities of geographic space that are not taken into account by these approaches. Particularly, concepts such as the hierarchical nature of geographic space and the topological relationships between the geographic objects must be considered in order to fully represent the relationships between the documents and to allow new and interesting types of queries to be posed to the system.

In this paper, we present an index structure that takes these issues into account. We first describe some basic concepts and related work in Section 2. Then, in Section 3, we present the general architecture of the system and describe its components. The system architecture defines a workflow for constructing a document database where both the words and the geographic references in the documents are considered. The core of the system architecture is an index structure that enables the system to store and access efficiently the documents using both their textual references and their geographic ones. Finally, the system architecture includes two different user interfaces: one for final users that can be used to pose queries to the system and to display the results, and another for system administrators that can be used to manage the document collections. After that, in Section 4, we describe some types of queries that can be answered with this system and we sketch the algorithms that can be used to solve this queries. Furthermore, Section 5 presents some experiments that we made to compare our structure with other ones that use a pure spatial index. Finally, Section 6 presents some conclusions and future lines of work.

2 Related Work

Inverted indexes are considered the classical text indexing technique. An inverted index associates to each word in the text (organized as a *vocabulary*) a list of pointers to the positions where the word appears in the documents. The set of all those lists is called the *occurrences* [1]. The main drawback of these indexes is that geographic references are mostly ignored because place names are considered words just like the others. If the user poses a query such as as *hotels in Spain*, the place name *Spain* is considered a word, and only those documents that contain that word are retrieved. A document containing only names of cities of Spain but not the exact word *Spain* is not retrieved by the system because it does not fulfil the textual query. Regarding indexing geographic information, many different spatial index structures have been proposed along the years. A good survey of these structures can be found in [9]. The main goal of spatial index structures is improving access time to collections of geographic data objects. One of the most popular spatial index structure and a paradigmatic example is the R-tree [10]. The R-tree is a balanced tree derived from the B-tree which splits space in hierarchically nested, possibly overlapping, minimum bounding rectangles. The number of children of each internal node varies between a minimum and a maximum. The tree is kept in balance by splitting overflowing nodes and merging underflowing nodes. Rectangles are associated with the leaf nodes, and each internal node stores the bounding box of all the rectangles in its subtree. The decomposition of space provided by an R-tree is adaptive (dependent on the rectangles stored) and overlapping (nodes in the tree may represent overlapping regions). A drawback of spatial index structures is that they do not take into consideration the hierarchy of space. Internal nodes in the structure are meaningless in the real world, they are just meaningful for the index structure. For example, imagine that we want to build an index for a collection of countries, provinces, and cities. These objects are structured in a topological relationship of containment, that is, a city is contained within a province that is itself contained within a country. If we build an R-Tree with these geographic objects, the containment hierarchy will not be maintained. The internal nodes of the R-Tree do not represent provinces or countries, and therefore, the hierarchy of space is not maintained in the index. It is not possible to associate some information to the node of a province and have the cities belonging to this province inherit this information because there is no relation at all between a province and its cities in the R-Tree index structure. Some work has been done to combine both types of indexes. The papers about the SPIRIT (Spatially-Aware Information Retrieval on the Internet) project [11–15] are a very good starting point. In [14], the authors conclude that keeping separate text and spatial indexes, instead of combining both in one, results in less storage costs but it could lead to higher response times. More recent works can be broadly classified into two categories depending on how they combine textual and spatial indexes. On the one hand, some proposal have appeared that combine textual and spatial aspects in an hybrid index [16, 17]. On the other hand, some proposals define structures that keep separate indexes for spatial and text attributes [6–8]. Our index structure is part of this second group because this division has many advantages [8]. Furthermore, in [7, 8], the authors survey the work in the SPIRIT project and propose improvements to the system and the algorithms defined. In their work they propose two naive algorithms: *Text-First* and *Geo-First*. Both algorithms use the same strategy: one index is first used to filter the documents (textual index in Text-First and spatial index in Geo-First), the resulting documents are sorted by their identifiers and then filtered using the other index (spatial index in Text-First and textual index in Geo-First). Nevertheless, none of these approaches take into account the relationships between the geographic objects that they are indexing.

A structure that can properly describe the specific characteristic of geographic space is an *ontology*, which is a formal explicit specification of a shared conceptualization [18]. An ontology provides a vocabulary of classes and relations to describe a given scope. In [19], a method is proposed for the efficient management of large spatial ontologies using a spatial index to improve the efficiency of the spatial queries. Furthermore, in [12, 15] the authors describe how ontologies are used in query term expansion, relevance ranking, and web resource annotation in the SPIRIT project. However, as far as we know, nobody has ever tried to combine ontologies with other types of indexes to have a hybrid structure that captures both the topological and the spatial relationships between the geographic objects indexed.

3 System Architecture

Fig. 1 shows our proposal for the system architecture of a geographic information retrieval system. The architecture can be divided into three independent layers: the index construction workflow, the processing services and the user interfaces. The bottom part of the figure shows the index construction workflow, which, in turn, consists of three modules: the document abstraction module (described in Section 3.1), the index construction module (the textual part of this process is described in Section 3.2 and the spatial part of this process is described in Section 3.3), and the index structure itself (described also in Section 3.3).

The processing services are shown in the middle of the figure. On the left side, the *Geographic Space Ontology Service* used in the spatial index construction is shown. This service is used extensively in the index construction module, and therefore it is described in Section 3.3. On the right side, one can see the two services that are used to solve queries. The rightmost one is the *query evaluation service*, which receives queries and uses the index structure to solve them. Section 4 describes the types of queries that can be solved by this service, as well as the algorithms that are used to solve these queries. The other service is a *Web Map Service* following the OGC specification [20] that is used to create

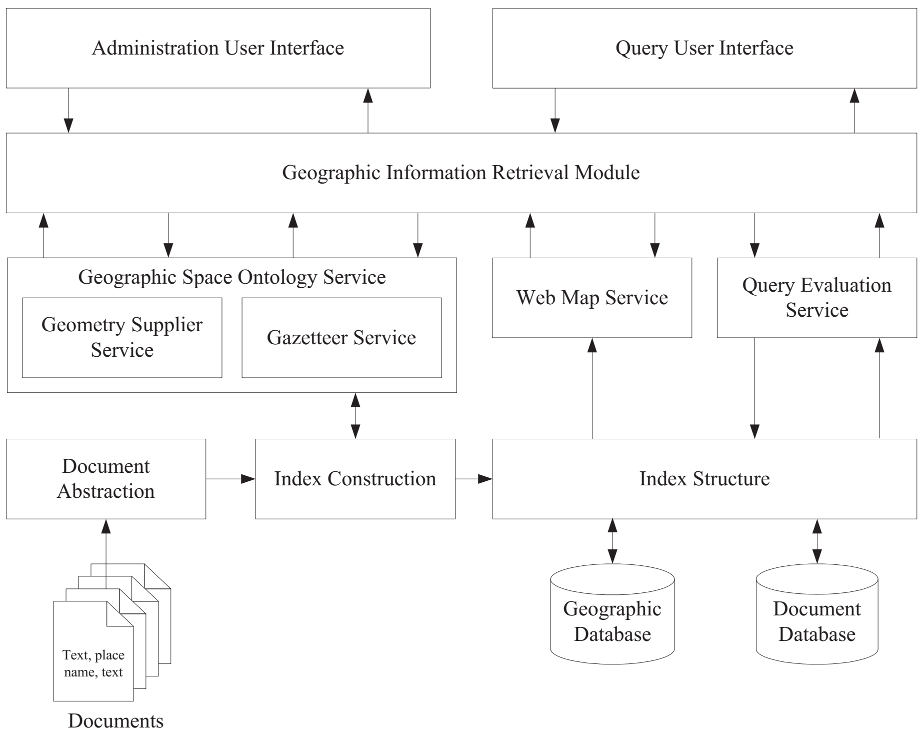


Fig. 1. System Architecture

cartographic representations of the query results. This service is not described in this paper. On top of these services a *Geographic Information Retrieval Module* is in charge of coordinating the task performed by each service to response the user requests. The topmost layer of the architecture shows the two user interfaces that exist in the architecture: the *Administration User Interface* and the *Query User Interface*. These user interfaces are described in Section 3.4.

3.1 Document Abstraction

Given that the system must be generic, it must support indexing several kinds of documents. These documents will be different not only because they may be stored using different file formats (plain text, XML, etc.), but also because their content schema may be different. A document collection may have a set of attributes that have to be stored in the index (such as *document id*, *author*, and *document text*), whereas other document collection may have a different set of attributes (such as *document id*, *summary*, *text*, *author*, and *source*).

To solve this problem, we have defined an abstraction that represents a *document* as a set of *fields*, each one with a value that is extracted from the document text. Each field can either be *stored*, *indexed*, or both. If a field is stored, its contents are stored in the index structure and they can be retrieved by a query. If a field is indexed, then this field is used to build the index structure. Furthermore, a field can be indexed textually, spatially, or in both indexes. The definition of a document as a set of fields is similar to the one used in the Lucene text search engine [21]. We have extended this idea adding the spatial indexing possibility.

In order to support different types of documents and different file formats, the document abstraction is exposed by the system as a programming interface that can be extended with particular implementations for different configurations of file formats and document schemas. In order to support a new configuration, a developer only has to implement the interface *DocumentFactory* that defines the operations that must be implemented in order to create *Documents*.

As an example for the validation of the system, we have indexed documents from the Financial Times collection [22]. The document collection is marked up in SGML (*Standard Generalized Markup Language*). Each document has a <DOCNO> tag including the TREC document identifier string and a <TEXT> tag including the main content of the document. Fig. 2 shows a partial example of a document in this collection.

```
<DOC>
     <DOCNO>FT941-6371</DOCNO>
     <TEXT>
          Senior European company executives are being invited to 'vote'
          for Europe's Most Respected Companies . . .
     </TEXT>
</DOC>
```

Fig. 2. Financial Times (TREC) example

To support this document collection, we defined a *TRECFTDocumentFactory* that builds documents with two fields. The first field contains the tag DOCNO content and it is stored but not indexed. The second field contains the tag TEXT content and it is not stored but indexed in both indexes.

3.2 Textual Indexing

As we said before, the index structure at the core of the system architecture contains both a textual index and a spatial index. We use Lucene [21] to implement a textual index. Lucene is a high-performance, full-featured text search engine library written entirely in Java. It is an open source project part of the Apache project. Lucene uses an object representation of the indexable documents. A *Document* in Lucene contains several *Fields*. A *Field* in Lucene is a pair (*name*, *value*) and information about whether it is stored and/or indexed. Field values are set using *Analyzers*. These analyzers implement several classical information retrieval techniques to reduce the number of indexed words and to improve the index performance such as *removing stopwords*, *stemmers*, etc. *StandardAnalyzer* is the most sophisticated analyzer built into Lucene's core. It is a parser with rules for email addresses, acronyms, hostnames, floating point numbers, as well as converting the value to lowercase and removing stop words.

In this stage of the workflow process, the system builds a Lucene index. Each of the documents built in the previous stage is inserted into the textual index. The document identifier is stored but not indexed in the textual index, and each field marked to be indexed in the textual index or in both indexes is indexed tokenized in the Lucene index but not stored.

3.3 Spatial Indexing

After building the textual index, the spatial index must be built. The spatial indexing is the most complex stage, and it comprises three steps. First, the system analyses the document fields that are marked as spatially indexable and extracts candidate location names from the text. In a second step, these candidate locations are processed in order to determine whether the candidates are real location names, and, in this case, to compute their geographic locations. There are some problems that can happen at this point. First, a location name can be ambiguous (*polysemy*). For instance, "*London*" is the capital of the United Kingdom and it is a city in Ontario, Canada too. Second, there can be multiple names for the same geographic location, such as "*Los Angeles*" and "*LA*". Finding geographical references in text is a very difficult problem and there have been some papers that deal with different aspects of this problem [6, 23, 24]. Web-a-where [23] uses "spatial containers" in order to identify locations in documents, MetaCarta (the commercial system described in [24]) uses a natural language processing method, and STEWARD [6] uses an hybrid approach. It is not the aim of this paper to deal with this problem but we describe how we obtain geographic references in order to complete the architecture description. Finally, the third step consists in building the spatial index with the geo-referenced locations computed in the previous step together with references to the documents containing them. We describe these three steps and the spatial index structure below.

Discovery of Location Names. For the discovery of candidate location names, all the spatially indexable fields are processed in order to discover the place names contained within. There are two *Linguistic Analysis* techniques that are widely used for this: *Part-Of-Speech* tagging and *Named-Entity Recognition*. On the one hand, Part-Of-Speech tagging is a process whereby tokens are sequentially labelled with syntactic labels, such as "verb" or "gerund". On the other hand, Named-Entity Recognition is the process of finding mentions of predefined categories such as the names of persons, organizations, locations, etc.

Our *Location Names Discovery* module uses the *Natural Language Tool LingPipe* [25] to find locations. It is a suite of Java libraries for the linguistic analysis of human language free for research purposes that provides both Part-Of-Speech tagging and Named-Entity Recognition. LingPipe involves the supervised training of a statistical model to recognize entities. The training data must be labelled with all of the entities of interest and their types. In the system validation with the Financial Times collection, we use LingPipe trained with the MUC6 corpus (http://www.ldc.upenn.edu) labelled with locations, people and organizations. After the LingPipe processing, the module filters the resultant named entities selecting only the locations and discarding people and organization names.

Geo-referenciation of Location Names. After discovering a collection of candidate location names, the system must distinguish false candidates and geo-reference the real ones. In this context, geo-referencing a location name implies not only to obtain its coordinates in a particular coordinate system, but also to obtain all the data needed to include the place in a spatial index. We have developed a system based on an ontology of the geographic space that is built using a *Gazetteer* and a *Geometry Supplier*.

A Gazetteer is a geographical dictionary that contains, in addition to location names, alternative names, populations, location of places, and other information related to the location. In our test implementation we use *Geonames* [26] that provides a geographical database available under a creative commons attribution license. This database contains more than two million populated places over the world with their latitude/longitude coordinates in WGS84 (*World Geodetic System 1984*). All the populated places are categorized so that it is possible to classify them into different administrative division levels (continents, countries, regions, etc.).

However, Geonames (and Gazetteers in general) does not provide geometries for the location names other than a single representative point. But for our spatial index we need the real geometry of the location name (for example, the boundary of countries). We defined a *Geometry Supplier* service to obtain the geometries of those location names. As a base for this service we used the *Vector Map* (VMap) cartography [27]. VMap is an updated and improved version of the National Imagery and Mapping Agency's Digital Chart of the World. It supplies first and second level administrative division geometries in a proprietary format. However, there are free tools that can create *shapefiles* from that format, such as FWTools [28]. We have created a PostGIS [29] spatial database with these shapefiles and we have done several corrections and improvements over this database.

Even though our test implementation uses Geonames and VMAP, it has been designed so that these components are easily exchangeable. All accesses to these components are performed through generic interfaces that can be easily implemented for other components.

This step combines both services in order to geo-reference location names. First, an ontology of the geographic space is defined. In our test implementation, the geographic space is divided into three levels of administrative divisions (continents, countries and regions) and a level of populated places. These four levels are organized in a hierarchical structure where each level geographically contains all objects in the next level.

Then, for each candidate location name, an *ontology path* must be built. This path will be used in the construction of the spatial index structure. For this task, a hierarchical structure following the design pattern *Chain of Responsibility* [30] was implemented. Fig. 3 shows a brief class diagram of this component. The structure is composed of four levels (continent, country, region, and populated place), one for each level of the ontology, but it is easily extensible. Each level contains a connection to the gazetteer and to the geometry supplier in order to retrieve the data needed by the process. Then, an algorithm in two steps obtains all possible geo-references associated with a location name. In the first step, each level obtains from the gazetteer all the locations with the requested name. If the gazetteer does not return any location for a given candidate location name, the candidate is discarded. In the second step, the system builds the complete ontology path from bottom to top. For instance, if the requested location name was London, in the first step the system obtains two locations with this name. After that, it returns the paths *United Kingdom*, *England*, *London* and *Canada*, *Ontario*, *London*.

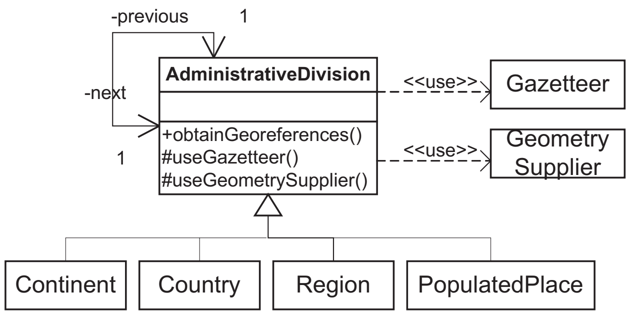


Fig. 3. Geo-references module

Spatial Index Construction. Fig. 4 shows a class diagram of the index structure. The main component of the index structure is a tree composed by nodes that represent location names. These nodes are connected by means of inclusion relationships (for instance, *Galicia* is included in *Spain*). The tree structure is built using the ontology paths computed by the process described in the previous section. In each node we store: (i) the keyword (a place name), (ii) the bounding box of the geometry representing this place, (iii) a list with the document identifiers of the documents that include geographic references to this place, and (iv) a list of child nodes that are geographically within this node. If the list of child nodes is very long, using sequential access is very inefficient. For this reason, if the number of children nodes exceeds a threshold, an R-Tree is used instead of a list.

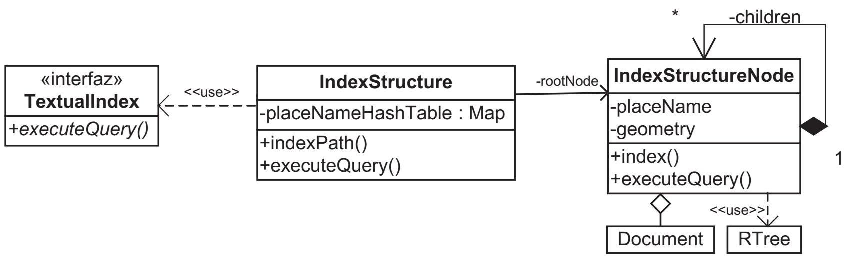


Fig. 4. Class diagram of the index structure

Two auxiliary structures are used in the index. First, a *place name hash table* that stores for each location name its position in the index structure. This provides direct access to a single node by means of a keyword that is returned by the Gazetteer Service if the word processed is a location name. The second auxiliary structure is the textual index with all the words in the documents that is used to solve textual queries (this index is described in section 3.2).

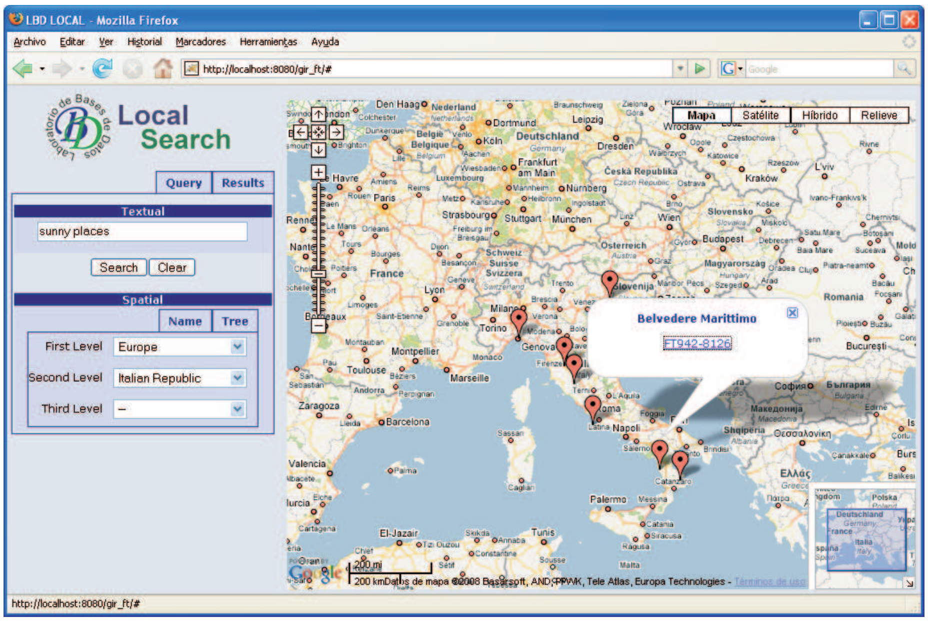
Keeping separate indexes for text and geographical scopes has many advantages. First of all, all textual queries can be efficiently processed by the text index, and all spatial queries can be efficiently processed by the spatial index. Moreover, queries combining textual and spatial aspects are supported. Furthermore updates in each index are handled independently, which makes easier the addition and removal of data. Finally, specific optimizations can be applied to each individual indexing structure.

However, this structure has two main drawbacks. First, the tree that supports the structure is possibly unbalanced penalizing the efficiency of the system. We present some experiments in Section 5. Our intention is to prove that it is not a very important problem. Second, ontological systems have a fixed structure and thus our structure is static and it must be constructed *ad-hoc.*

3.4 User Interfaces

The system has two different user interfaces: an administration user interface and a query user interface. The administration user interface was developed as a stand-alone application and it can be used to manage the document collection. The main functionalities are: creation of indexes, addition of documents to indexes, loading and storing indexes, etc. The main screen of this interface shows useful information about the loaded index such as the number of documents indexed, the fields of each of these documents, the number of location names in the index, etc.

Fig. 5 shows a screenshot of the query user interface. This interface was developed as a web application using the *Google Maps API* [31]. This API provides a number of utilities for manipulating maps and adding content to the map.


LBD LOCAL - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
http://localhost:8080/gir_ft/#
Laboratorio de Bases de Datos
Local Search
Query
Results
Textual
Search
Clear
Spatial
Name
Tree
Mapa
Satélite
Híbrido
Relieve
Belvedere Marittimo
FT942-8126

sunny places

First Level: Europe

Second Level: Italian Republic

Third Level: --

Fig. 5. Query User Interface

In the next section we sketch the types of queries that can be solved with this system. These queries have two different aspects: a textual aspect and a spatial aspect. The query user interface allows the user to indicate both aspects. The spatial context can be introduced in three ways that are mutually exclusive:

– *Typing the location name.* In this case, the user types the location name in a text box. This is the most inefficient way because the system has to obtain all geo-references associated with the typed place name and it is a time-expensive process.
– *Selecting the location name in a tree.* In this case, the user sequentially selects a continent, a country within this continent, a region within the country, and a populated place within the region. If the user wants to specify a location name of a higher level than a populated place, it is not necessary to fill in all the levels. The operation is very easy and intuitive because the interface is implemented with a custom-developed component using the AJAX technology that retrieves in the background the location names for the next level. When the user selects a place in the component, the map on the right zooms in automatically to the selected place.

– *Visualizing the spatial context of interest in the map.* The user can navigate using the map on the right to select the spatial context of interest. The system will use the bounding box of this map as the query window if the user did not type a place name or did not select a location name.

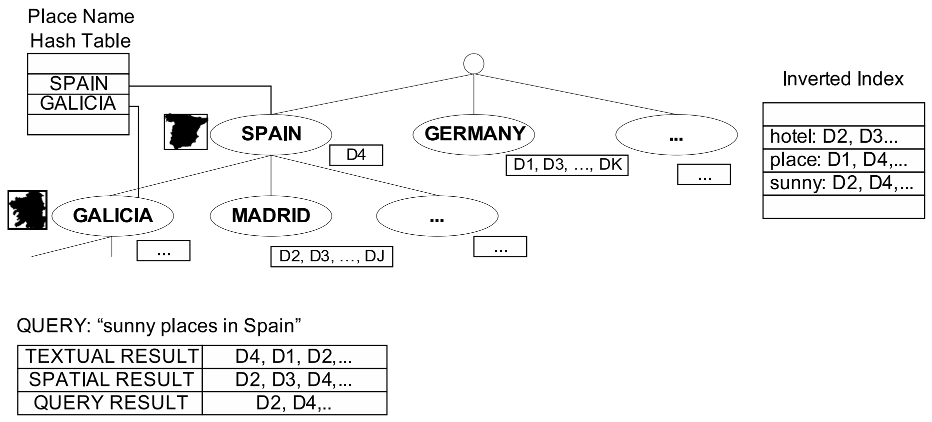
4 Supported Query Types

The most important characteristic of an index structure is the type of queries that can be solved with it. The following types of queries are relevant in a geographic information retrieval system:

– *Pure textual queries.* These are queries such as "*retrieve all documents where the words* hotel *and* sea *appear*".
– *Pure spatial queries.* An example of this type of queries is "*retrieve all documents that refer to the following geographic area*". The geographic area in the query can be a point, a query window, or even a complex object such as a polygon.
– *Textual queries with place names.* In this type of queries, some of the words are place names. For instance, "*retrieve all documents with the word* hotel *that refer to Spain*".
– *Textual queries over a geographic area.* In this case, a geographic area of interest is given in addition to the set of words. An example is "*retrieve all documents with the word* hotel *that refer to the following geographic area*".

Inverted indexes can solve pure textual queries by retrieving from the inverted index the lists of documents associated to each word and then performing the intersection of the lists. Pure spatial queries can be solved by spatial indexes by descending the structure and taking into consideration only those nodes whose bounding box intersects with the geographic area of the query. This operation returns a set of candidate documents that has to be refined with the actual geographic reference in order to decide whether the document is part of the result or not.

Pure textual queries can be solved by our system because a textual index is part of the index structure. Similarly, pure spatial queries can also be solved because the index structure is built like a spatial index. Each node in the tree is associated with the bounding box of the geographic objects in its subtree. Therefore, the same algorithm that is used with spatial indexes can be used with our structure. However, the index structure that we propose can be used to solve the third and fourth types of queries, which cannot be easily solved using a textual index and a spatial index. For the case of the query with place names, our system can discover that *Spain* is a geographic reference by querying the Gazetteer service and then we can use the *place name hash table* in the structure to retrieve the index node that represents *Spain.* Thus, we save some time by avoiding a tree traversal. Fig. 6 shows how these type of queries can be solved by the index structure. The textual index of the structure can be seen on the right part of the figure, whereas the spatial index can be seen on the left part. When the user poses a query with the text *sunny places* and the place name *Spain*, the textual index is used to retrieve the list of documents that contain the words, and the index structure is used to compute the list of documents that reference the geographic area. These two lists can be seen at the bottom part of the figure. Then, the result to the query is computed as the intersection of both lists.



TEXTUAL RESULT	D4, D1, D2,...
SPATIAL RESULT	D2, D3, D4,...
QUERY RESULT	D2, D4,..

Fig. 6. Example of the index structure

Regarding the fourth type of query, the textual index is used to retrieve the list of documents that contain the words and the ontology-based index structure is used to compute the list of documents that reference the geographic area. Then, the intersection of both lists is the result to the query. We analyze the performance of our structure to solve this type of queries in comparison with other proposals using a pure spatial index in Section 5. The conclusion of these experiments is that *the performance of our structure is acceptable in comparison with index structures using pure spatial indexes.*

Another improvement over text and spatial indexes is that our index structure can easily perform query expansion on geographic references because the index structure is built from an ontology of the geographic space. Consider the following query "*retrieve all documents that refer to Spain*". The query evaluation service will discover that Spain is a geographic reference and the place name index will be used to quickly locate the internal node that represents the geographic object *Spain*. Then all the documents associated to this node are part of the result to the query. Moreover, all the children of this node are geographic objects that are contained within Spain (for instance, the city of Madrid). Therefore, all the documents referenced by the subtree are also part of the result of the query. The consequence is that the index structure has been used to expand the query because the result contains not only those documents that include the term *Spain*, but also all the documents that contain the name of a geographic object included in Spain (e.g., all the cities and regions of Spain).

5 Experiments

In the previous section we showed that our structure has a qualitative advantage over systems that combine a textual index with a pure spatial index because query expansion can be performed directly with our index structure (e.g. *retrieve all documents with the word* hotel *that refer to Spain*). Hence, our index structure supports a new type of query that cannot be implemented with a pure spatial index. However, unlike pure spatial index structures, our index structure is not balanced and therefore, the query performance can be worse.

In this section we describe the experiments that we performed to compare our structure with other ones based on a pure spatial index. We used the TREC FT-91 (Financial Times, year 1991) document collection [22], which consists of 5,368 news documents. Then, we built two indexes over this collection: one using our index structure as described in this paper, and another one using a textual index and an R-Tree [10]. Furthermore, we developed an algorithm to generate random spatial query windows. This algorithm is based on the performance comparisons of the R*-Tree in [32] and it generates query windows where the ratio of the x-extension to the y-extension uniformly varies from 0.25 to 2.25 and the centres of the query rectangles are uniformly distributed all over the world. Fig. 7 shows several query windows generated using this algorithm.

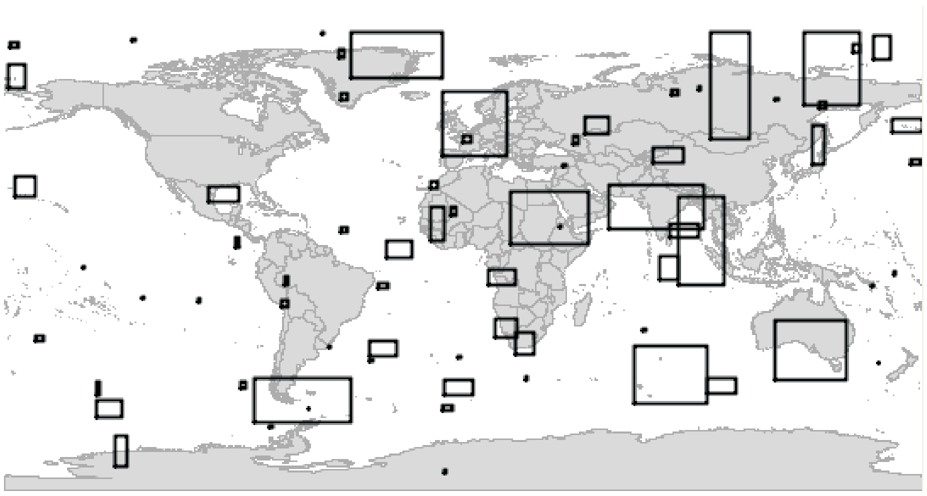


Fig. 7. Example of randomly generated query windows

We compared the structures with respect to four different query window areas, namely 0.001%, 0.01%, 0.1% and 1% of the world. We generated 100,000 random query windows for each area, and we averaged the computing time of each query execution. Table 1 shows the results of this experiment.

Table 1. Ontology-based index versus R-Tree

Query area	0.001%	0.01%	0.1%	1%
Our index	0.013	0.017	0.052	0.360
R-Tree	0.010	0.016	0.057	0.370

The first row of the table shows the results obtained with our structure (in milliseconds), and the second one shows the results obtained with the structure using an R-Tree. Both index structures have similar performance. The performance of our structure is a bit worse than the R-Tree when the query window is small but, surprisingly it is a bit better than the R-Tree when the query window is bigger. In order to explain this surprising result, we analyzed the performance in particular zones. We distinguished two relevant types of zones and we repeated the experiment generating random queries in both zones. First, we studied the performance of the structures when the document density is high (see Table 2). In this case, the performance of our structure is higher than the R-Tree performance. We believe this is because our structure stores a list of documents for each location while the R-Tree uses a node for each one document.

Table 2. Ontology-based index versus R-Tree (zones of high document density)

Query area	0.001%	0.01%	0.1%	1%
Our index	0.03	0.11	1.05	9.84
R-Tree	0.07	0.22	1.64	12.85

Second, we studied the performance when the documents density is low (see Table 3). In this case, the R-Tree performance is better because the number of nodes in both structures is similar and the R-Tree is balanced whereas our structure may be unbalanced. For this reason, in the general case, when the query window is small the probability of that query window being in a high document density zone is small and, therefore, the R-Tree performance is better. However, when the query window is bigger that probability is higher and, therefore, the R-Tree performance is lower.

Table 3. Ontology-based index versus R-Tree (zones of low document density)

Query area	0.001%	0.01%	0.1%	1%
Our index	0.02	0.03	0.09	0.4
R-Tree	0.02	0.03	0.07	0.2

6 Conclusions and Future Work

We have presented in this paper a system architecture for an information retrieval system that takes into account not only the text in the documents but also the geographic references included in the documents and the ontology of the geographic space. This is achieved by a new index structure that combines a textual index, a spatial index and an ontology-based structure. We have also presented how traditional queries can be solved using the index structure. Finally, new types of queries that can be solved with the index structure are described and the algorithms that solve these queries are sketched.

Future improvements of this index structure are possible. We are currently working on the evaluation of the performance of the index structure, particularly we are performing experiments to determine the precision and recall. Moreover, *Toponym Resolution* techniques must be implemented to solve ambiguity problems when we geo-reference the documents. Another line of future work involves exploring the use of different ontologies and determining how each ontology affects the resulting index. Furthermore, we plan on including other types of spatial relationships in the index structure in addition to inclusion (e.g., adjacency). These relationships can be easily represented in the ontology-based structure, and the index structure can be extended to support them. Finally, it is necessary to define algorithms to rank the documents retrieved by the system. For this task, we must define a measure of spatial relevance and combine it with the relevance computed using the inverted index.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley (1999)
2. Worboys, M.F.: GIS: A Computing Perspective. CRC (2004) ISBN: 0415283752.
3. ISO/IEC: Geographic Information – Reference Model. International Standard 19101, ISO/IEC (2002)
4. Open GIS Consortium, Inc.: OpenGIS Reference Model. OpenGIS Project Document 03-040, Open GIS Consortium, Inc. (2003)
5. Global Spatial Data Infrastructure Association: Online documentation. Retrieved May 2007 from http://www.gsdi.org/.
6. Lieberman, M.D., Samet, H., Sankaranarayanan, J., Sperling, J.: STEWARD: Architecture of a Spatio-Textual Search Engine. In: Proceedings of the 15th ACM

Int. Symp. on Advances in Geographic Information Systems (ACMGIS07), ACM Press (2007) 186 – 193

7. Chen, Y.Y., Suel, T., Markowetz, A.: Efficient query processing in geographic web search engines. In: SIGMOD Conference. (2006) 277–288
8. Martins, B., Silva, M.J., Andrade, L.: Indexing and ranking in Geo-IR systems. In: GIR '05: Proceedings of the 2005 workshop on Geographic information retrieval, New York, NY, USA, ACM Press (2005) 31–34
9. Gaede, V., Günther, O.: Multidimensional access methods. ACM Comput. Surv. **30**(2) (1998) 170–231
10. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In Yormark, B., ed.: SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984, ACM Press (1984) 47–57
11. Jones, C.B., Purves, R., Ruas, A., Sanderson, M., Sester, M., van Kreveld, M., Weibel, R.: Spatial information retrieval and geographical ontologies an overview of the SPIRIT project. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (2002) 387 – 388
12. Jones, C.B., Abdelmoty, A.I., Fu, G.: Maintaining ontologies for geographical information retrieval on the web. In: Proceedings of On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE Ontologies, Databases and Applications of Semantics, ODBASEŠ03. Volume 2888 of Lecture Notes in Computer Science. (2003)
13. Jones, C.B., Abdelmoty, A.I., Fu, G., Vaid, S.: The SPIRIT Spatial Search Engine: Architecture, Ontologies and Spatial Indexing. In: Proceedings of the 3rd Int. Conf. on Geogrphic Information Science. Volume 3234 of Lecture Notes in Computer Science. (October 2004) 125 – 139
14. Vaid, S., Jones, C.B., Joho, H., Sanderson, M.: Spatio-Textual Indexing for Geographical Search on the Web. In: Proceedings of the 9th Int. Symp. on Spatial and Temporal Databases (SSTD). Volume 3633 of Lecture Notes in Computer Science. (2005) 218 – 235
15. Fu, G., Jones, C.B., Abdelmoty, A.I.: Ontology-Based Spatial Query Expansion in Information Retrieval. In: Proceedings of In On the Move to Meaningful Internet Systems 2005: ODBASE 2005. Volume 3761 of Lecture Notes in Computer Science. (2005) 1466 – 1482
16. Zhou, Y., Xie, X., Wang, C., Gong, Y., Ma, W.Y.: Hybrid index structures for location-based web search. In: CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management, New York, NY, USA, ACM (2005) 155–162
17. Hariharan, R., Hore, B., Li, C., Mehrotra, S.: Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems. In: Proceedings of the 19th Int. Conf. on Scientific and Statistical Database Management (SSDBM07), IEEE Computer Society (2007)
18. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition **5**(2) (June 1993) 199 – 220
19. Dellis, E., Paliouras, G.: Management of Large Spatial Ontology Bases. In: Proceedings of the Workshop on Ontologies-based techniques for DataBases and Information Systems (ODBIS) of the 32nd International Conference on Very Large Data Bases (VLDB 2006). (September 2006)
20. Open GIS Consortium, Inc.: OpenGIS Web Map Service Implementation Specification. OpenGIS Project Document 01-068r3, Open GIS Consortium, Inc. (2002)

21. Apache: Lucene. Retrieved October 2007 from http://lucene.apache.org.
22. National Institute of Standards and Technology (NIST): TREC Special Database 22, TREC Document Database: Disk 4. Retrieved November 2007 from http://www.nist.gov/srd/nistsd22.htm.
23. Amitay, E., Har'El, N., Sivan, R., Soffer, A.: Web-a-where: geotagging web content. In: SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (2004) 273–280
24. Rauch, E., Bukatin, M., Baker, K.: A confidence-based framework for disambiguating geographic terms. In: Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references, Morristown, NJ, USA, Association for Computational Linguistics (2003) 50–54
25. Alias-i: LingPipe, Natural Language Tool. Retrieved October 2007 from http://www.alias-i.com/lingpipe/.
26. Geonames: Gazetteer. Retrieved September 2007 from http://www.geonames.org.
27. National Imagery and Mapping Agency (NIMA): Vector Map Level 0. Retrieved September 2007 from http://www.mapability.com.
28. FWTools: Open Source GIS Binary Kit for Windows and Linux. Retrieved September 2007 from http://fwtools.maptools.org.
29. Refractions Research: PostGIS. Retrieved June 2007 from http://postgis.refractions.net.
30. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley (1996)
31. Google: Google Maps API. Retrieved November 2007 from http://www.google.es/apis/maps/.
32. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. SIGMOD Rec. **19**(2) (1990) 322–331