Proposal of an Ontology Based Web Search Engine

Adrián Ponce, Claudia Deco, Cristina Bender

Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario
Av. Pellegrini 250, 2000 Rosario, Argentina
adrianvp@gmail.com, deco@fceia.unr.edu.ar, bender@fceia.unr.edu.ar

Abstract

When users search for information in a web site, sometimes they do not get what they want. Assuming that the scope where the search take place works fine, there are some problems caused by the way the user interact with the system, others that refer to characteristics of the language used, and others caused by the lack or nonexistent semantics in web documents. In this work, we propose a web search engine of a particular web site that uses ontologies and information retrieval techniques. Although the architecture we propose is applicable to any domain, the experimentation was done in a tourism web site. The results show a substantial improvement in the effectiveness of the search, with a gain of 33% in Precision.

Keywords: ontologies, web information retrieval

1. INTRODUCTION

Nowadays, many web sites handle a big amount of information. These web sites usually have a search engine that allows users to locate and access information in a fast and direct way, without having to browse all the web site. However, sometimes users do not get what they want. These unfruitful searches can happen because of a set of factors that affect search results.

One of the problems found is this scenario is that different documents can use different words to refer to a same concept, and therefore, when keyword search is done, the system will not retrieve all documents that refer to a same concept. These problems are caused by synonyms and it is called synonymy of terms. Moreover, the input terms of a query can have multiple meanings, and search engines can not interpret what the user wanted to search, retrieving non-relevant documents. This problem is called polysemy of terms. Other problem appears in large document collections, where a big amount of results is retrieved and it is essential that the engine order them by some kind of ranking [1]. However, this order cannot coincide with the user's expectations, and a possible desired result can be relegated because of a lower ranking. Studies show that most users do not look more than the 100 first results. Moreover, the first 30 are considered the most important [2].

The objective of this work is to improve a web search engine using Semantic Web tools and Information Retrieval techniques for searching a web site. The goal is to solve and/or reduce the problems mentioned earlier and consequently to improve the effectiveness of the search. For this purpose, we introduce a controlled vocabulary to reduce synonymy and polysemy, stemming to normalize the vocabulary used in the system and therefore increasing the amount of retrieved documents, and lastly we improve the search precision with the use of ontologies. To evaluate our proposal we implement a prototype applied to a tourism web site as a specific domain. We verify search improvements measuring its effectiveness in terms of Precision and number of retrieved documents.

2. BASIC CONCEPTS

Information Retrieval deals with representation, storage, organization and access to information items [3]. Information Retrieval systems perform searches over a collection of documents written in natural language [4]. The goal is to retrieve information from the collection, with the aim to satisfy the user's information needs.

The documents are indexed by the terms that they contain. The process of generation, construction and storing documents representations is called indexing and as a result, we obtain inverted indexes [5]. An inverted index allows fast access to documents that contains a specific term. For this purpose, it maintains a register for each term in the document collection, which in its simplest form consists in the term name and the list of documents where that term occurs. For the inverted index generation usually non-significant words (stopwords) are first eliminated. In addition, inflectional and derived forms of a word are reduced to a common form. This last process is called stemming [6]. Stopwords are removed during text analysis of documents and from user queries, because they do not add significant information to the search. The stemming technique helps to normalize the vocabulary of the information retrieval system, increasing the number of retrieved documents for a user query.

To make easy the retrieval of particular documents, metadata can be used. Metadata are structured data that describe characteristics of information entities and help in the identification, discovery and manipulation of those entities [7]. Metadata information ought to belong to a controlled vocabulary. A controlled vocabulary or thesaurus is a list of words and phrases carefully selected. It solves the problem caused by synonymy and polysemy forcing each concept to be described by a unique allowed term. Consequently, they reduce the ambiguity of natural language and ensure consistency of the vocabulary [8].

The evaluation of an information retrieval system is based on the notion of relevant and non-relevant documents. A document is relevant, if the user perceives that it contains information of value with respect to his information needs [1]. The typical indicators to measure the effectiveness in information retrieval are Precision and Recall. Precision is the fraction of retrieved documents that are relevant. Recall is the fraction of relevant documents that are retrieved. When it comes to web searching, Precision is measured over fixed amounts of first retrieved documents. For Recall computation, the total amount of relevant documents retrieved and non-retrieved is needed. However, in the web is impossible to know the total amount of relevant documents that exists. So it is often used the amount of retrieved documents as an alternative measure. We will refer to it here as Recall*.

A method for enhancing search performance is query expansion [9]. Query expansion is the process of adding new terms to a given user query, in an attempt to provide better contextualization and the hope to retrieve documents, which are more useful to the user [3]. During the expansion, new terms are added, replaced or even discarded. These terms can be extracted from dictionaries, thesaurus or ontologies that can be dependant or not from the corpus of documents.

An ontology is an explicit specification of a conceptualization [10]. A conceptualization is an abstract and simplified vision of the world (or domain) that when it is specified, a formal description is done. This allows that ontologies can be interpreted by machines and can make reasoning about them. Ontologies consist in a set of classes, relations, instances and axioms. Classes represent concepts that belong to the domain that describes the ontology. Relations represent an association between elements of the ontology. Instances are used to represent particular elements of a class. Axioms are statements that are assumed to be true. The Web Ontology Language (OWL) is the language for writing ontologies recommended as standard by the World Wide Web Consortium (W3C). It is divided in three types: OWL Lite, OWL DL and OWL Full;

each one provides different levels of semantics expressivity. OWL Lite allows the construction of taxonomy of classes, express equalities, inequalities and simple restrictions. OWL DL allows maximum expressivity, maintaining computational completeness and decibility of the reasoning. OWL Full has the maximum expressivity but there are no computational guaranties.

3. SEARCH ARCHITECTURE

The proposed search architecture is shown on Figure 1. It consists on a web search engine, an inverted index, an ontology and a collection of web documents. These web documents are annotated with metadata extracted from the ontology.

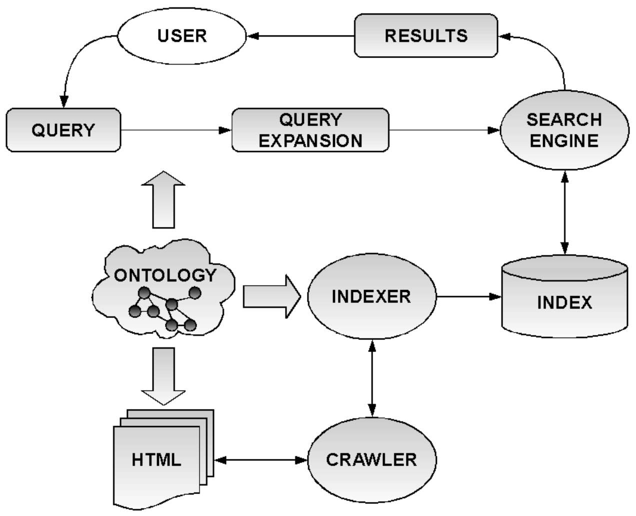

Figure 1: Search architecture

A web search engine has three main components: a crawler, an indexer and a search engine. The crawler collects information from an annotated collection and stores it in a local repository. The indexer processes that repository and generates an inverted index from it, using a controlled vocabulary extracted from the ontology. The input queries are expanded with information extracted from the ontology, and then, they are sent to the search engine. This engine uses the inverted index to answer the query and it returns the results to the user.

The ontology is specific to the web site application domain. In this work, a tourism web site from Argentina is used. This ontology is used for web documents annotation process, for index generation and for query expansion. Web documents annotation process inserts additional information (metadata), to facilitate the retrieval of documents being annotated. For this purpose, documents are classified according to the ontology taxonomy. The classification process is hard to automate. In this work, we have decided to do it manually. The difficulty is that web documents must be classified by their underlying content and not by terms that they contain. Once the document was classified, it is annotated with metadata that consist in a list with the name of the ontology classes, subclasses and possibly individuals that classify the document. The annotated metadata is indexed under a common field, which allows querying the index for documents that contain a particular value in that field. This information is used during the query expansion to retrieve documents and to improve precision.

To reduce synonymy, user query terms and documents terms are restricted, to the ones that belong to a controlled vocabulary. Its terms consist of ontology named classes, because they designate the concepts used in the domain. The restriction is done replacing each term that can be associated to an ontology concept, by the controlled vocabulary term, which designates that concept. In the case of user queries, the replacing takes place during query expansion, whereas in the case of documents, during index generation (the preferred term is indexed). The lists of words that can be associated to a concept are maintained by the developer and are inserted into the ontology by means of annotation properties. For example, given the ontology class *Accommodation,* which determines the controlled vocabulary term *Accommodation*, could be annotated by the words: *hostel*, *hotel*, *inn, camping*, etc. The use of a controlled vocabulary and stemming has by collateral effect the lost of exact queries. To deal with this problem, it is decided the indexing of all document terms under other index field to query it, in the case of exact queries.

The query expansion strategy consists in two steps. In the first step, the input words are replaced by the controlled vocabulary terms, whenever is possible. In a second step, related terms generated from the ontology are added to the possible modified query. The aim of these related terms is to retrieve documents that deal about the associated concepts with the query terms and to improve the ranking of most important documents under consideration. For this purpose, the document annotated with metadata is searched by those terms, because they classify a document by the underlying concepts.

3.1 Ontology Design

The ontology was created with Protégé-OWL version 3.3.1 (protege.stanford.edu/) ontology editor. To define the tourism domain taxonomy, concepts were identified and then, they are represented by means of classes. These classes are organized in the hierarchy shown on Figure 2.

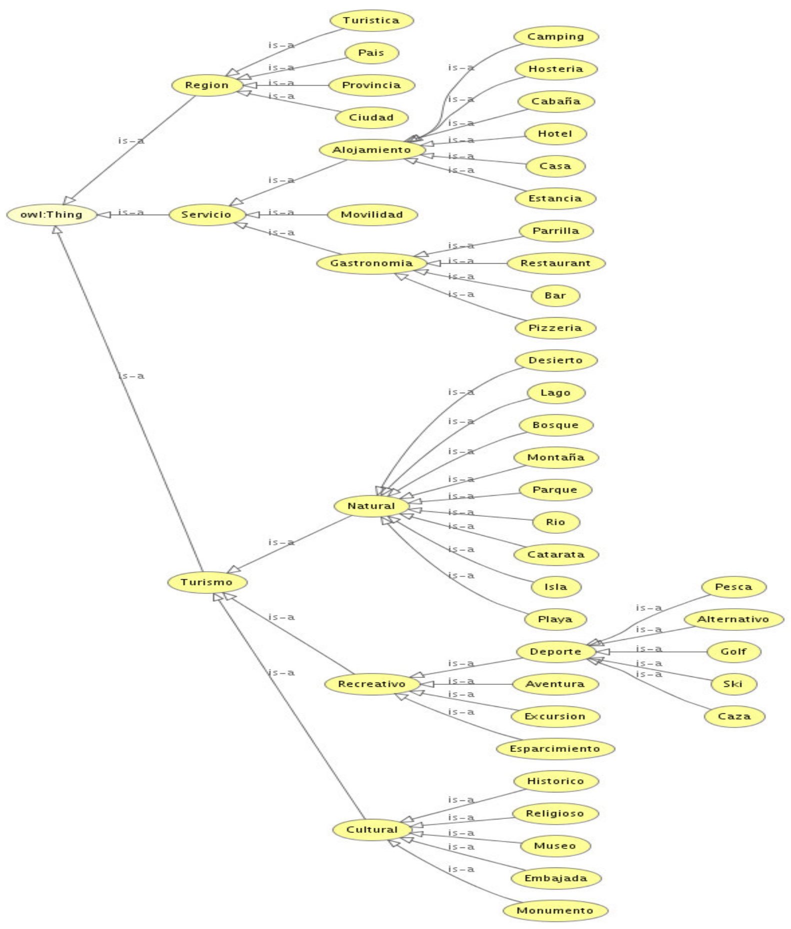

Figure 2: Tourism Ontology

In this ontology, sibling classes are disjoint to assure that an individual can belong to only one of them. In OWL, the properties represent binaries relations over individuals. The object properties are associations between individuals. The annotation properties can be used to annotate information (metadata) into classes, individuals and properties. The added information can be a literal, an URI or an individual as well. In the Tourism ontology, we use the annotation property "term" to add information by literals of type XMLSchema#string into the classes. The objective is to add lists of equivalent terms to the ones that belong to the controlled vocabulary.

In Figure 3 the graphic of classes, properties and restrictions that belong to the Tourism ontology. Classes are represented by rectangles and the properties by arrows. The asterisks at the side of a property name represent multiple cardinality.

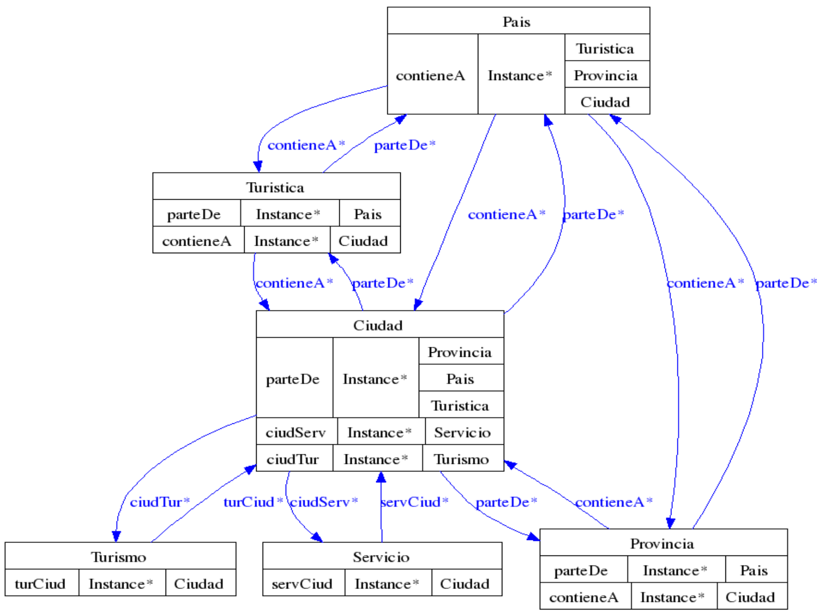

Figure 3: Classes, properties and restrictions

Individuals are used to do a more accurate classification of particular documents and lately annotate them with respect to this classification. Moreover, when they are combined with the query expansion, allows the retrieval of more accurate information. In this ontology, 450 individuals were inserted.

3.2 Query Expansion

When a user submits a query, the query expansion process begins. Firstly, the user input query is transformed by the system into a Nutch syntax query, before applying the query expansion process. Secondly, keyword substitution and stemming is done, and a Lucene engine syntax search expression is generated using Nutch’s default approach for searching. Lastly, additional query terms are added from ontological information associated to the substituted input query. The additional terms are used to query the category field of the inverted index, with the goal of retrieving those documents that deal about concepts associated with the input query terms and also to increase the relevance of particular retrieved documents. The process is shown on Figure 4.

These additional terms are obtained from the input query terms that can be associated with an ontology class or individual, and the application of an algorithm for the query processing that uses a particular heuristic. This heuristic has the goal of obtaining additional information from the input query terms.

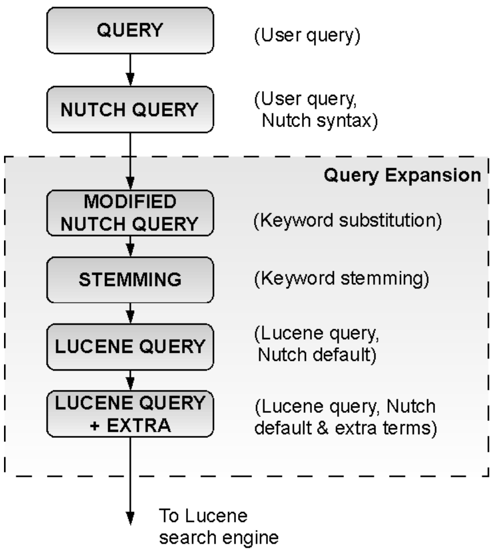

Figure 4: Query Expansion

To this purpose, it can reason using the ontology considering the following cases:

– There is only one input query term that can be associated to a class or individual: then the class or the individual name is returned respectively.
– The input query terms are associated to two classes (i.e. class1 and class2 in that order):
  i) If a class (i.e. class1) is subclass of the other (class2), it returns the name of the first one (class1), because it is more specific than the other is, and therefore, it is considered as a more accurate information.
  ii) If none of them is subclass of the other, and exists relationships of type R(x_i,y_i) or R'(y_i,x_i) such that individuals x_i and y_i are instances of class1 and class2 respectively, then the name of those x_i will be returned. This is because generally, leftmost query search terms are more important (in this case class1). Otherwise, both classes names are returned, because more accurate information cannot be obtained from the query terms. For example, if the associated classes are *City* and *State*, and if there exists a *part-of* relationship between their individuals, then the heuristic will return the *cities* names that are part of the *state*.
– The query terms are associated with a class and an individual:
  i) If the individual is an instance of the class, the heuristic returns the individual name because in this case the class name would be redundant information. For example, if the class is *Country* and the individual *Argentina* (*Country instance*), then Argentina is returned, because it can be inferred from the ontology that *Argentina* is a *Country*.
  ii) If the individual is not an instance of the class, and exists relationships of type R(x_i,individual) and R'(individual,x_i) such that individuals x_i are instance of the class, then the heuristic returns the name of those x_i; otherwise returns null. For example, if the class is *City*, the individual is *Argentina* and R is the relationship *part-of*, then the names of cities that are part of Argentina would be returned.
– The query terms are associated with two individuals (i.e. individual1 and individual2 in that order): if there is a relationship between both individuals, it returns individual1, because it is considered that generally the leftmost term of the query search is the most important; otherwise, it returns null.

Query expansion example:

Let us suppose that the user input query is “locations of the state of Misiones”. The prototype first removes all the Stopwords and adds a “+” to the remaining terms, to specify that they are required terms:

```
+location +state +misiones
```

Then it replaces the keywords of the default search field (content) when it is possible, and applies the stemming algorithm. Assuming that *city* is the term of the controlled vocabulary associated with the word *location*, then the following expression constitutes the Nutch default approach for searching:

```
+(url:locat anchor:locat content:cit title:locat host:locat)
+(url:stat anchor:stat  content:stat title:stat host:stat)
+(url:mision anchor:mision content:mision title:mision host:mision)
url:"locat stat mision"~1000
anchor:"locat stat mision"~4
content:"cit stat mision"~1000
title:"locat stat mision"~1000
host:"locat stat mision"~1000
```

The top three clauses, that are required, consult the url, anchor, content, title and host index fields, and determine the set of documents to rank. The default conjunction operator is OR. This also applies to the expressions of type *<field>:<value>* that are inside brackets, and therefore, every document that satisfy some of those expressions will be retrieved for ranking. The last five clauses are proximity search clauses that are not required (i.e. without “+”). Their only goal is to increase the ranking of retrieved documents, when the searched terms are separated by a certain distance (notated as ~1000 and ~4). The distance is the maximum number of terms by which the query search terms can be found separated inside a document.

The next step is to add additional terms that are generated from the ontology. The final query sent to the search engine is showed in Figure 5.

```
1 | +(url:locat anchor:locat content:cit title:locat host:locat
  | category:city category:state category:pmisiones)
2 | +(url:provinci anchor:stat content:stat title:stat host:stat
  | category:city category:state category:pmisiones)
3 | +(url:mision anchor:mision content:mision title:mision host:mision
  | category:city category:state category:pmisiones)
4 | url:"locat stat mision"~1000
5 | anchor:"locat stat mision"~4
6 | content:"cit stat mision"~1000
7 | title:"locat stat mision"~1000
8 | host:"locat stat mision"~1000
9 | category:eldorado category:obera category:posadas category:puertoiguazu
```

Figure 5: Final query for the example

In the example query, the classes *City*, *State* and the individual that corresponds to the State of Misiones, *pmisiones*, are associated to the query terms, and the terms *category:city*, *category:state* and *category:pmisiones* are generated. These terms are added to each required clause (lines 1, 2 and

3 in Figure 5). Now the set of retrieved documents will include those that are annotated with such metadata. At the end, it is added new clauses generated from heuristic returned terms, that impact only in the ranking of retrieved documents. In other words, it is added line 9 in Figure 5. This increases the relevance of retrieved documents that deals about cities, which are part of the state of Misiones.

Nutch lets the use of boost factors for each query field (not shown for easier the example understanding). These factors give a measure of the relevance assigned to each field in the moment of ranking computation. In this work, it is considered that the category fields are more relevant than the others. Therefore, they are assigned a greater boosting factor. This is because a category metatag characterizes a document by its content and not only by the occurring terms. Category terms that appear in the required clauses are assigned a boost factor value of 5, whereas for non required clauses it is assigned a value of 1000. With this last boost factor, it is accomplished a considerably impact over document scoring, and therefore, increasing document ranking.

4. RELATED WORK

In [11], in the retrieval model used, terms vectors are ontology instances instead of words that belong to the document vocabulary. Several heuristics are applied during query expansion, with the aim of dealing with ontologies imperfections, such as lack of semantics, ambiguity terms, incomplete ontologies, etc. Then, for each one of the heuristics, independent queries are created and sent to the search engine. Finally, the result is generated from combining results from independent queries. The metadata are generated automatically from the ontology during the index generation, and therefore, they are not inserted into the documents.

OWLIR [12] is a system designed for the retrieval of documents that contains either plain text or semantics tags in RDF and OWL. The system takes text documents as input, annotates them using the ontology and then indexes them. It performs full text search and field search. Moreover, it can infer additional semantics relationships, using the ontology and document metadata.

Swoogle [13] is an information retrieval system for RDF and OWL documents that reside in the web. It is designed for the automatic discovering of such documents (with a web crawler), indexing document metadata and answering queries about them. Moreover, the system presents interfaces for interacting with web services, people and software agents.

In [14] an architecture for the retrieval of individual document sections that belong to a particular domain is presented. Metadata are used for the identification of such sections, allowing the user to perform structured searches from a predefined set of categories that are maintained in an ontology. The system consists in an indexer, an ontology, and a relational database engine. The indexer represents documents in XML syntax and compares them to predefined categories in the ontology. When matching occurs, a database table register is created. To search for information, user queries are translated first to SQL, and then are solved in the relational engine.

5. PROTOTYPE IMPLEMENTATION

To implement the proposed architecture, the web search engine Nutch[1] was chosen. It is based on the Apache Lucene information retrieval library, and it allows to extend its functionalities by means of plug-ins. Nutch is free and it is developed in Java, which implies portability to the most known operating systems. The interaction with the ontology was implemented using Jena2[2] framework

[1] Nutch, Open Source Search: http://lucene.apache.org/nutch/

[2] Jena2, A Semantic Web Framework for Java: http://jena.sourceforge.net/

ontology API, in conjunction with the SPARQL[3] query language. For the reasoning process it was used the Jena2 reasoners and its inference API. The Spanish stemmer was implemented from the Apache Lucene and Snowball[4] projects. The stemming algorithm is a Porter-like[5] stemming algorithm, and it is used during index generation and during the user query processing.

The metadata processing, the index generation with controlled vocabulary, the query expansion process and the Spanish stemmer were implemented as Nutch plug-ins. Moreover, two new functionalities were added to the user graphic interface, with the aim of helping the user during the search task. One of them is the "Related search" field. With this option, the user can perform new searches from information inferred from the ontology with respect to the input query. The other functionality is given by the "Other search" field, with the goal of dealing with problems that are introduced by ambiguity terms. These ambiguity problems appear when the system tries to associate an ontology resource with a keyword or phrase, when there is more than one candidate for this association. To this purpose, the system shows the user a set of dropdown lists that allows him to reformulate the query and therefore, to perform a more accurate query.

6. EXPERIMENTATION

The experimentation was performed in a tourism web site of Argentina. The ontology was built from this web site. A corpus of 2500 manually annotated web pages was prepared.

For evaluation purpose, Precision at 10, 30 and 50 results were registered and the amount of retrieved documents (Recall*), over a set of 10 queries. These queries were solved using different configurations, to see effects on the system with the introduction of a particular functionality. Table 4 shows Precision and Recall* for the web search engine working with the default configuration (Nutch); with controlled vocabulary restriction (Nutch VC); with the use of Spanish stemming algorithm (Nutch+Stem); with the simultaneous use of controlled vocabulary and stemming (Nutch+VC+Stem); and lastly, when it uses query expansion (Nutch+QE).

In general, it is observed that there is very low increase in the amount of retrieved documents with Nutch VC. This indicates that the vocabulary used in documents is consistent with respect to the input query terms.

There are not noticeable changes between Nutch and Nutch VC, when it comes to Precision.

	Nutch				Nutch VC				Nutch + Stem				Nutch + VC + Stem				Nutch + QE			
User Query	Precision 10	30	50	Recall*	Precision 10	30	50	Recall*	Precision 10	30	50	Recall*	Precision 10	30	50	Recall*	Precision 10	30	50	Recall*
gaucho	0.40	0.47	0.28	30	0.40	0.47	0.28	30	0.80	0.60	0.62	51	0.80	0.60	0.62	51	0.80	0.60	0.62	51
deporte	0.80	0.80	0.80	68	0.80	0.80	0.80	68	1.00	0.63	0.54	1164	1.00	0.63	0.54	1164	1.00	1.00	1.00	1171
transporte	0.70	0.30	0.20	34	0.70	0.47	0.30	70	0.30	0.27	0.18	81	0.50	0.43	0.30	77	1.00	0.80	0.56	92
ciudad argentina	0.50	0.53	0.34	1118	0.40	0.57	0.34	1224	0.30	0.47	0.34	1078	0.30	0.47	0.36	1185	1.00	1.00	1.00	1197
turismo "san carlos de bariloche"	0.60	0.50	0.36	41	0.60	0.50	0.36	41	0.80	0.53	0.36	41	0.80	0.53	0.36	41	1.00	0.67	0.58	477
museo litoral	0.50	0.23	0.14	50	0.50	0.23	0.14	50	0.20	0.27	0.16	117	0.20	0.27	0.16	117	0.80	0.27	0.16	172
turismo aventura argentina	0.80	0.43	0.30	1047	0.80	0.43	0.30	1047	0.80	0.43	0.34	1046	0.80	0.43	0.34	1046	1.00	1.00	0.92	1379
atractivo natural argentina	0.40	0.40	0.30	34	0.40	0.40	0.30	34	0.30	0.27	0.26	106	0.30	0.23	0.26	107	1.00	1.00	1.00	155
región turística argentina	0.90	0.30	0.18	190	0.90	0.30	0.18	219	0.70	0.30	0.18	281	0.60	0.30	0.18	372	1.00	0.33	0.20	424
excursión región litoral argentina	0.20	0.07	0.04	3	0.40	0.13	0.08	10	0.00	0.07	0.04	23	0.00	0.13	0.08	30	0.40	0.53	0.34	218
Average	**0.58**	**0.40**	**0.29**	**261.50**	**0.59**	**0.43**	**0.31**	**279.30**	**0.52**	**0.38**	**0.30**	**398.80**	**0.53**	**0.40**	**0.32**	**419.00**	**0.90**	**0.72**	**0.64**	**533.60**

Table 4: Experimentation results

The use of Stemming (Nutch+Stem) increases considerably the amount of retrieved documents. It is observed a slightly decrease of Precision for some searches, and a slightly gain for others. Both

[3] SPARQL Query Language for RDF: http://www.w3.org/TR/rdf-sparql-query/
[4] Snowball, a language for stemming algorithms: http://snowball.tartarus.org/
[5] Porter Stemming Algorithm: http://tartarus.org/~martin/PorterStemmer/index.html

configurations Nutch+VC+Stem and Nutch+Stem increase the amount of retrieved documents with respect to Nutch default. In terms of Precision, there is not a clear trend for this indicator.

The configuration Nutch+QE increases the amount of retrieved documents more than other configurations, because it includes those documents that refer concepts associated with user query terms, and therefore there can be documents in which the query terms do not occur. It is also observed a significant increase in the search Precision at 10, 30 and 50 results, achieving 100% of Precision for some cases. When no concept is associated to the input query terms as in the query "gaucho", Nutch+QE presents the same values of Precision and Recall* that are obtained for Nutch+VC+Stem. This is because there is no concept/individual that belongs to the ontology that can be associated with the query term "gaucho", which means that during query expansion, the system does not add terms of type "category" that either increases the amount of retrieved documents or improves Precision.

Figure 6 shows the average amount of retrieved documents for each configuration. For (Nutch+QE), it is observed a Recall* value that doubles the one obtained with Nutch default. Figure 7 shows the average Precision values for each system configuration at 10, 30 and 50 results.

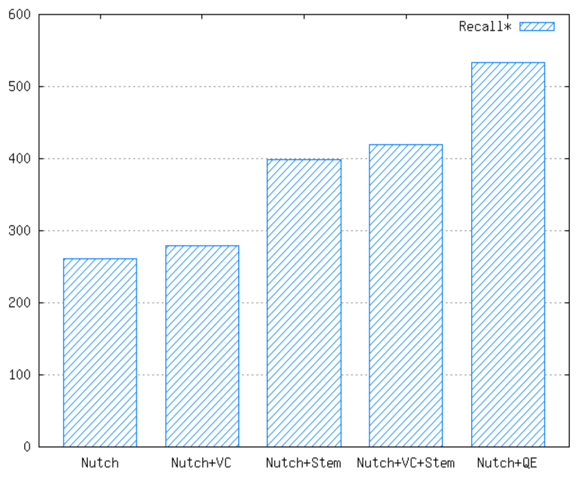

Figure 6: Recall*

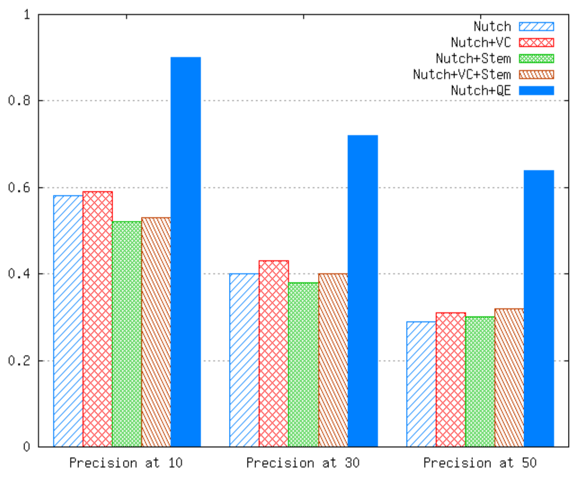

Figure 7: Precision at 10, 30 y 50

In the three cases, Nutch with query expansion (Nutch+QE) presents a Precision increase of about 33% with respect to Nutch default. In summary, the combination of Nutch with query expansion shows higher Precision and Recall* values with respect to Nutch default. In other words, this means that the effectiveness of searches is improved with the proposed architecture.

7. CONCLUSIONS

Experimentation results show a substantial improvement of search effectiveness, with a gain of near 33% in Precision and a Recall* of about the double with respect to the Nutch default behavior.

The use of a controlled vocabulary, transparently improves the user experience, because it hides specific domain knowledge from the user. It also maintains the consistency of the vocabulary used in every document that is part of the web site, which leads to reduce synonymy and polysemy. Furthermore, this technique and stemming help to regularize the system vocabulary, and increase the amount of retrieved documents. The metadata classifies web documents, categorizing them with the use of the ontology's taxonomy. This semantics enrichment allows the retrieval of documents that can be considered as non-relevant by a conventional platform. The ontology was used for the definition of the controlled vocabulary, generation of the metadata, during the query expansion, and to deal with ambiguity terms.

More and more information is handled by web sites. Therefore, keyword searching that offers most of information retrieval systems sometimes is not enough. The introduction of ontology semantics, not only can improve the information retrieval effectiveness, but also lets the organization and comprehension of a web site content, to the point that, this content can be automatically processed by machines or software agents.

As future work, we propose the automatic document annotation, the reuse of existent ontologies to describe the application domain. Another issue is the use of a lexical database instead of the table of equivalent terms associated to the controlled vocabulary terms. As a complement during the query expansion process, a relevance feedback module can be added.

REFERENCES

[1] C. Manning, P. Raghavan, H. Schutze. “An Introduction to Information Retrieval”. Cambridge University Press, UK, 2007.

[2] M. Busby. “Learn Google: All About Search Engines”. Wordware Publishing, 2003.

[3] R. Baeza-Yates, B. Ribeiro-Neto. “Modern Information Retrieval”. Addison Wesley, UK, 1999.

[4] E. Voorhees. “Natural Language Processing and Information Retrieval”. In Information Extraction: Towards Scalable, Adaptable Systems, pp.32-48, Springer, Germany, 1999.

[5] A. Singhal. "Modern information retrieval: A brief overview". Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 24, no. 4, pp. 35-42, 2001.

[6] M. F. Porter. “Snowball, a Language for Stemming Algorithms”, October 2001: http://snowball.tartarus.org/texts/introduction.html

[7] American Library Association, Task Force on Metadata. Summary Report, June 1999: http://www.libraries.psu.edu/tas/jca/ccda/tf-meta3.html

[8] F. W. Lancaster. “El Control del Vocabulario en la Recuperación de Información”. Ed. Universidad de Valencia, España, 1995.

[9] E. N. Efthimiadis. "Query Expansion". Annual Review of Information Science and Technology (ARIST), vol. 31, pp. p121-87, 1996.

[10] T. R. Gruber. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing". International Journal Human-Computer Studies Vol. 43, Issues 5-6, p.907-928, 1995

[11] G. Nagypàl. "Improving Information Retrieval Efectiveness by Using Domain Knowledge Stored in Ontologies". FZI Research Center for Information Technologies at the University of Karlsruhe, pp. 780-789, Germany, 2005.

[12] U. Shah, T. Finin, A. Joshi, J. May_eld, and R. S. Cost. “Information Retrieval on the Semantic Web”. In Proceedings of the Eleventh International Conference on Information and Knowledge Management, pp 461-468, New York, USA, 2002.

[13] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. “Swoogle: a Search and Metadata Engine for the Semantic Web”. In Proceedings of the Thirteenth ACM international Conference on information and Knowledge Management, pp. 652-659, New York, USA, 2004.

[14] S. R. El-Beltagy, A. Rafea, Y. Abdelhamid. “Using Dynamically Acquired Background Knowledge for Information Extraction and Intelligent Search”. In Intelligent Agents for Data Mining and Information Retrieval, Idea Group Publishing, pp. 195-206, USA, 2004.