Expert Systems with Applications 41 (2014) 2886–2896

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

NLP-based faceted search: Experience in the development of a science and technology search engine

Marcelo G. Armentano *, Daniela Godoy, Marcelo Campo, Analia Amandi

ISISTAN Research Institute (CONICET-UNICEN), Campus Universitario, Paraje Arroyo Seco, Tandil 7000, Argentina

ARTICLE INFO

Keywords:
Vertical search engines
Faceted search
Named entities recognition
Natural language processing

ABSTRACT

An appropriate promotion, distribution and dissemination of scientific, artistic and technology developments can foster the collaboration between a country's productive and academic sectors. The purpose of this paper is to present a novel search engine aiming at helping people to access science and technology advances, researchers and institutions working in specific areas of research. Our search engine first collects information disseminated on the Web in academic institution sites and in researchers personal homepages. Then, after intensive text processing, it summarizes the information in an enriched and user-friendly presentation oriented to non-expert users. Stable performance and an acceptable level of effectiveness for automatic named entities recognition indicate the potential of our approach for bridging the gap between the heterogeneous and unstructured information available on the Web about the research and development advances in a country and the innovation required by the productive sectors.

1. Introduction

Information about science and technology (S& T) advances of a country, including articles, patents and technological developments, is often available from official websites of S& T agencies, national and private universities, research institutes and other institutions (Bar-Ilan, 2000). For researchers, practitioners, entrepreneurs and the general public it might be difficult to access this corpus of knowledge scattered on the Web, hindering possible interactions between the parties and preventing fruitful collaboration (Wong and Yap, 2012).

In this paper, we describe our experience on the development of *SciTechSE*, a search engine specialized on S& T information that was tested in our country. The main purpose of the project was to bring scientific knowledge to people in order to foster collaboration with the country productive sectors. The result was a specialized S& T search engine[1] that indexes pages on Web servers belonging to scientific and educational institutions nationwide with the goal of promoting the distribution and dissemination of scientific, artistic and technological developments in the country.

Instead of creating and maintaining a centralized database, the main purpose of *SciTechSE* is to take advantage of the information already published by researchers and institutions in their Web sites. Then, it replaces the effort of loading and manually organizing such knowledge, which implies to augment the workload of researchers, with an automatic information extraction method applied to the text of Web pages. This method recognizes interesting entities, such as individuals, institutions and places using natural language processing (NLP) techniques.

Furthermore, it was required to present information to non-expert users in a friendly interface, allowing them to rapidly identify people, institutions and places related with their information needs. This requirement leads to the use of *faceted search* as a mechanism to organize Web search results. Hierarchical faceted categories (HFC) (Hearst, 2006) are built and populated using the extracted entities. Each category corresponds to a different facet, dimension or feature type relevant to the collection of Web pages to be navigated. Thus, facets help to identify categories of interest so that users can interactively filter results to find pages meeting their needs.

SciTechSE is the first search engine specialized in science and technology bounded to our country. Other existing search engines specialized in the same domain mainly act as a gateway to science and technology documents from reliable scientific Websites with structured information. The search engine presented in this article works on the unstructured information published in the websites of the institutions as well as in the homepages of the researchers.

The rest of this paper is organized as follows. Section 2 present some related work. Section 3 presents the search engine, whereas Section 4 describes the general architecture of the proposed tool. Section 5 describes some experiments conducted

* Corresponding author. Tel.: +54 2494439682.
E-mail address: marcelo.armentano@isistan.unicen.edu.ar (M.G. Armentano).
[1] Available online at http://serverhp.isistan.exa.unicen.edu.ar:8080/buscador/.

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.10.023

to evaluate the performance of the search engine in identifying relevant entities. Finally, Section 6 states conclusions drawn from this experience.

2. Related work

In the last years, many efforts have been made to parse text documents and discovery useful knowledge from them (Patterson et al., 2008; Jacquenet and Largeron, 2009; Hong et al., 2010; Choi, 2011). SciTechSE can be framed within the research on vertical search engines. Vertical search engines differ from general search engines in that they focus on a specific industry, topic, type of content (e.g., travel, movies, images, blogs, live events), piece of data, geographical location, etc. This kind of search engines are useful to find content that might be difficult to find on a general search engine due to the great amount of documents that are indexed by the later.

Tang et al. (2006) compared the performance of domain-specific health and depression search engines with a general-purpose engine (Google) on both relevance of results and quality of advice. They conducted a standard IR experiment, running queries on engines, pooling the results for each query, and employing research assistants to judge them. Authors found that domain-specific search might provide more relevant results, since it indexes a non-uniformly chosen, relevant subset. Domain specific search might also provide higher quality results, if its subset includes high-quality information sources and avoids pages with false, harmful or misleading information.

Several search engines focused on science and technology have been developed. Course Homepage Finder (Altingovde et al., 2007), for example, maintains and queries course homepages for various university departments. This prototype system first crawls the web seeded with the Turkish university homepages and restricted to .edu domains. Then, an information extraction engine is trained to extract course ids, course names, semester, instructors' names and emails, and store them in a relational DBMS. Scitopia.org[2] is a free federated vertical search service which retrieves content provided by its twenty-one partner scholarly societies. Scitopia.org acts as a gateway to the research most cited in scholarly work and patents, searching more than 3.5 million documents from the leading electronic libraries in major science and technology disciplines. Results to a query are presented to the user organized in different clusters (Topics, Authors, Publications, Publishers, Affiliations and Years). Science.gov[3] is an integrated single-search gateway to reliable science and technology information from over 2100 scientific Websites. Science Research[4] is a free search engine allowing access to numerous scientific journals and public science databases. Similar initiatives are Scitation[5] and TechXtra.[6]

Most of the previously cited approaches are limited to offering a gateway to a set of authoritative sources of structured information about documents. Differently, SciTechSE collects unstructured information that is already present on the Web in academic institution sites and researchers personal homepages and, after intensive text processing, summarizes the information in an enriched and user-friendly presentation oriented to non-expert users.

Regarding faceted navigation, several works have been also presented. Prasad and Guha (2008) addressed the issue of concept naming vs. concept categorization by introducing the semantic annotating mechanism with facet analysis and coordination techniques. Yee et al. (2003) presented an interface for large image collections that allows users to navigate explicitly along conceptual dimensions that describe the images. The interface uses hierarchical faceted meta-data and dynamically generated query previews integrating category browsing with keyword searching. Images in the collection contained standard arts meta-data facets, including artist names, types of media, and dates, but it lacked meta-data categories that describe the appearance of items or the images depicted in them. To solve this problem, authors developed an algorithm to semi-automatically convert these descriptions into a set of meta-data categories. This was done by comparing the words in the descriptions to their higher-level category labels in WordNet (Fellbaum, 1998), and retaining a subset of the most frequently occurring categories. Ramirez and Mattmann (2004) presented the Automatic Concept Extraction (ACE) algorithm, which can aid users performing searches using search engines. Concepts of underlying web page results being searched are presented to the user in order to reduce the amount of links the user will need to visit to find her desired results. The ACE algorithm is based on emphasizing a concept through the amount of times it appears on a web page and emphasizing a concept using HTML tags.

Dakka and Ipeirotis (2008) presented an approach for automatic identification of facets that can be used to browse a collection of free-text documents. The technique presented builds on the idea that external resources, when queried with the appropriate terms, provide useful context that is valuable for locating the facets that appear in a database of text documents. As external resources for providing context authors used WordNet, Google, and Wikipedia. These resources are queried to examine which terms tend to co-occur often with the terms from the indexed documents under the assumption that facet terms are rare terms in the original database but co-occur frequently in the external resources with the terms that appear in the original database. Authors apply an expansion procedure, in which the important terms from each news story are expanded with "context terms" derived from the external resources. The expanded documents then contain many of the terms that can be used as facets. Sleiman and Corchuelo (2013) presented an unsupervised information extractor that works on two or more web documents generated by the same server side template. It finds the relevant information that should be extracted from them by removing shared token sequences among those documents.

The difference of the above mentioned approaches with our approach is that we focus on helping people to access researchers and institutions working in specific areas of research. For this reason, predefined facets are selected to identify People, Organizations and Places that can be used to refining search results.

3. SciTech search engine overview

SciTech search engine was created with the goal of assisting the population of our country to access the scientific developments in the country, supporting searches both in Spanish and English. Since the search engine is oriented to non-expert users, most of the effort was dedicated to providing a friendly user interface. Because of this reason, the graphical interface of SciTechSE was entirely developed using Google Web Toolkit (GWT)[7] that enables rich Web application development in Java. The interface communicates with SciTechSE's front-end to send queries, parse XML responses, etc.

Fig. 1 shows a screenshot of SciTechSE's main page. We can observe in this figure that the user entered the query "artificial intelligence". The response of the search engine is the set of pages containing those terms along with the set of people, institutions and places automatically identified in these pages using NLP techniques. We can also observe in label "11" of Fig. 1 that the user has

[2] http://www.scitopia.org/scitopia/.
[3] http://www.science.gov/.
[4] http://www.scienceresearch.com/scienceresearch/.
[5] http://scitation.aip.org/.
[6] http://www.techxtra.ac.uk/.
[7] http://code.google.com/webtoolkit/.

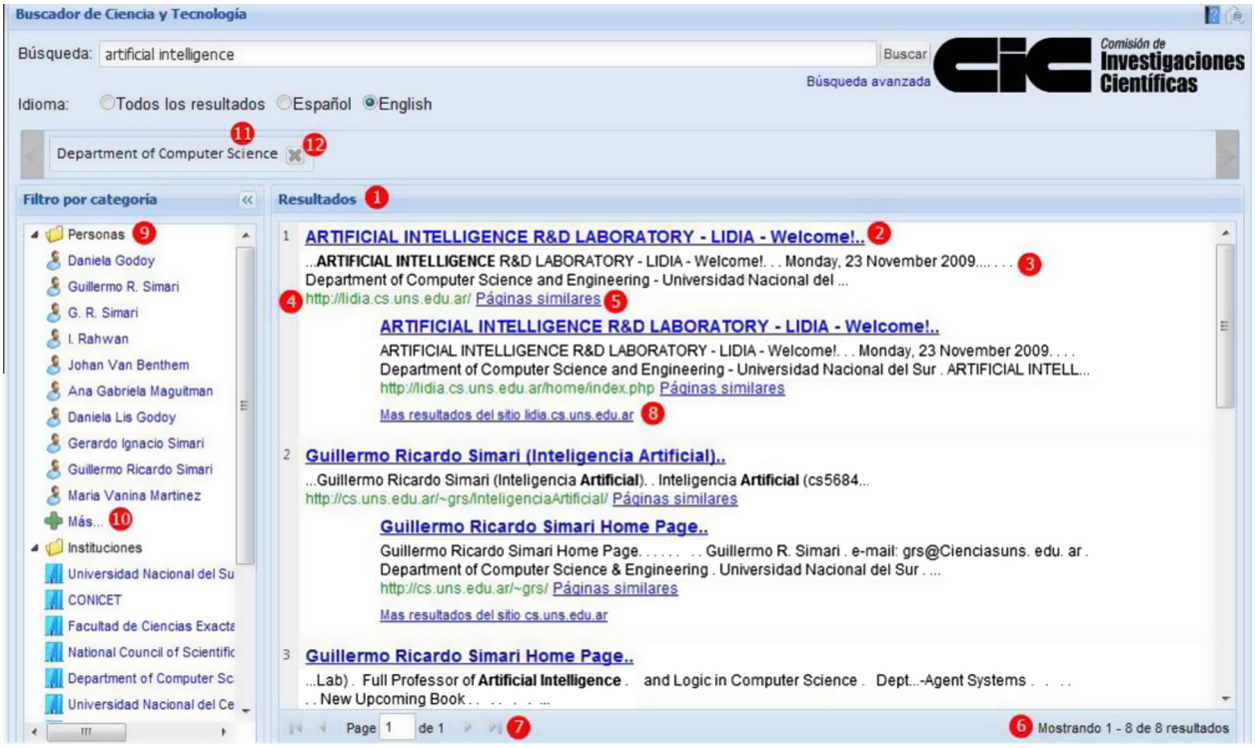

Fig. 1. SciTechSE screenshot showing the main page and search Web results.

selected "Department of Computer Science" from the set of institutions presented as facets so that the search results are reduced to those pages in which that specific entity was recognized. In the remaining of this section we detail the main features available in SciTechSE.

Users post queries to SciTechSE as if it was a traditional search engine, using a set of keywords. As a result, SciTechSE presents Web pages matching those terms in the results pane (labeled with "1" in Fig. 1), paged with 20 results per page.

For each result, SciTechSE shows its title (labeled with "2" in Fig. 1), an extract of the document in which the searched terms appeared (labeled with "3" Fig. 1) and the Web address of the result (labeled with "4" in Fig. 1). At the bottom of the results panel the total number of retrieved Web pages is shown (labeled with "6" in Fig. 1 and a navigation panel that paginate results (labeled with "7" Fig. 1).

Some other additional features of SciTechSE include:

- **More-Like-This** is used to find documents similar to a document returned by the search engine. Suggestions are made when the user clicks on a link named "Similar Pages" (labeled with "5" in Fig. 1). To find these similar documents SciTechSE uses the vectors of terms stored in the index. More information about this feature can be found in Section 4.3.2.
- **Highlighting** is the ability to highlight the text that matches the query in the information of the results displayed to the user (snippets). Terms used in the search query are highlighted in bold font, as can be seen in Fig. 1.
- **Field Collapsing** is a feature that allows users to group results with the same value for a certain field. The field used by SciTechSE to implement this feature is the site or domain of the retrieved pages. This causes that documents belonging to the same domain that a previously retrieved document (with a higher ranking score) will be removed from the response. The user can then see all the collapsed results using the link named "more results from this site . . . " (labeled with "8" in Fig. 1).
- **Spell Checking** provides spelling suggestions of the words entered by the user in the form of "Did you mean: . . . ". To implement this feature, the query is processed, analyzers are applied and then it is contrasted with the dictionary of terms in the index. More information about this feature can be found in Section 4.3.1.
- **Faceted Navigation** improves the search results with additional information about the retrieved pages. The browser interface is enriched with facets that can then be used as filters in a subsequent query. These facets correspond to different entities recognized in the text. The entities identified are presented in three different categories (People, Organizations and Places) on the left of the search results (labeled with "9" in Fig. 1). Notice that only ten entities are shown in each category. The set of all entities identified can be viewed by clicking on the link "More. . . " found at the end of each category (labeled with "10" in Fig. 1). By selecting one or more of these entities it is possible to restrict the results to only those pages or documents containing the selected entities. After applying a filter, it is visually located the filters bar (labeled with "11" in Fig. 1), between the search box and results. The user can apply as many filters as desired by clicking on other entities or delete a previously applied filter by clicking on the cross that is located next to it in the filters bar (labeled with "12" in Fig. 1). The filters applied are cumulative, meaning that every time a new filter is applied the number of results obtained will be equal to or less than the results presented without the application of the new filter. Facets are sorted on the interface according to this value. More information about faceted search can be found in Section 4.3.3.

4. SciTech architecture

The general architecture of SciTechSE is similar to most Web search engines, having three major elements: Web crawling, indexing and searching. Fig. 2 depicts the search engine general architecture. First, the crawler visits Web sites and follows links

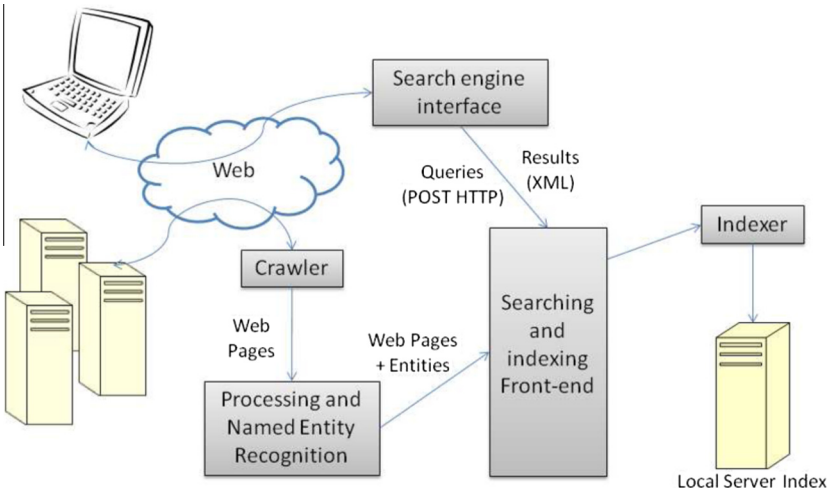

Fig. 2. General architecture of SciTech search engine.

to other sites looking for Web pages. It also returns to them on a regular basis to look for changes. Every page visited by the crawler goes to a central repository that is indexed for fast retrieval and ranking. Finally, users search the indexed information specifying a query that reflects their information needs. User queries are compared against the index to retrieve a subset of pages satisfying the user need and the resulting pages are ranked according to their relevance to the query.

In the SciTechSE search engine the crawler was implemented using Apache Nutch.[8] It periodically connects to those Web pages in which it is interested in. The visited pages are then processed to extract the entities contained in the text and, finally, an index is created to provide an efficient search service to end users. Since SciTechSE specializes in the domain of scientific and technological sites in our country, Web sites that are currently analyzed are institutional sites of universities and scientific and technological institutes explicitly given as seeds for crawling. The crawler starts with the default web page for each of those seeds, extracts the web links contained in those pages and traverse these links up to a predefined depth whenever the linked page fulfill a set of inclusion and exclusion rules. These rules aim at keeping the process limited to the domain we are interested in. For example, the rule `+ĥttp://([a-z0–9]*)*edu.ar` accepts all pages within the edu.ar domain. Notice that by changing the seeds and the crawling rules, SciTechSE can be adapted to work with institutions of other countries.

The Web pages that are recovered by the crawling process are analyzed searching for names, organizations and places that will be then presented to the user in the form of facets in the search engine interface. The user can interact with these facets to filter the results obtained from a given query in order to narrow down the results. In the following section we detail the process we follow to analyze the text present in the crawled Web pages in order to obtain the mentioned entities.

4.1. NLP and named entity recognition

Named Entity Recognition (NER) is a sub-task of the information retrieval process that aims at identifying atomic elements in the text and classifying them into predefined categories such as names, places, organizations or addresses, among other elements of interest for a particular domain. For this task, the text is processed using linguistic techniques together with statistic ones. SciTechSE NER module is based on GATE,[9] an open source text processing engine, which was adapted and extended to recognize names, organizations and places both in English and Spanish constrained to the domain of science and technology. Named entity recognition involves three steps: statement segmentation, part-of-speech (POS) tagging and NER itself. Each of these steps are detailed in the following sections.

4.1.1. Sentence segmentation

Sentence identification is done using a rule-based method that looks for potential sentence delimiters. These rules identify dot characters as sentence delimiters but take into account common exceptions such as acronyms and dots within numbers. Sentence segmentation is done in three steps. First, text is tokenized. Second, rules are applied to identify sentence delimiters. Third, start-of-sentence and end-of-sentence tags are added to the text.

The rules for sentence segmentation are configured by three files containing regular expressions, one regex per line. The three different files encode patterns for:

- *internal splits*: sentence splits that are part of the sentence, such as sentence ending punctuation;
- *external splits*: sentence splits that are not part of the sentence, such as 2 consecutive new lines;
- *non splits*: text fragments that might be seen as splits but they should be ignored, such as full stops occurring inside abbreviations.

NLP rules for sentence segmentation assume well-formed text. However, web pages usually contain text fragments without punctuation such as those in web forms, different kind of lists and tables. To solve this problem we implemented a set of heuristics that transform web text into text suitable to be used by the rule-based method for sentence segmentation. For example, certain HTML tags, such as < br/>, were also considered as sentence delimiters. We also appended missing a full stops to the text contained within tags such as < p>,<li>, <td>, <h1>, <h2>, etc.

4.1.2. Part-of-speech tagging

Part-of-speech (POS) tagging, also known as grammatical tagging, is a process that needs to be applied before analyzing the content of the sentences identified in the previous step. This pro-

[8] http://nutch.apache.org/.

[9] http://gate.ac.uk/.

cess assigns to every word in the sentence a grammatical category (noun, verb, article, etc.). POS tagging is done using both the word itself and the context in which the word is written, that is to say the position of the word with respect to adjacent words in the phrase, sentence or paragraph.

Most words in a text correspond to nouns and adjectives. For example, a pattern to identify a person in the text *"Perez, J. A. Jr."* could be "*a proper noun followed by a comma, one or more initials and an optional suffix*". For each entity that need to be identified in the text there will be a set of patterns defined to discover those entities based on the tags assigned by the POS tagging process.

The tagger used in SciTechSE is based on the Brill tagger (Hepple, 2000), which produces a part-of-speech tag as an annotation on each word or symbol. The tagging accuracy reported for this algorithm is 97.05%. We consider eight basic POS tags: noun, verb, adjective, article, adverb, conjunction, preposition and pronoun. There are further specific tags for each basic tag. For example, for nouns the POS tagger identifies different forms such as singular, plural and possessive. The POS tagger used by SciTechSE identifies forty-five tags in total.

4.1.3. Named entities recognition

In this step of the process, different entities are identified and classified into three different categories: persons, organizations and places.

Rules for named entities recognition (NER) were defined for English and Spanish and each group is applied after a language identification step. Both sets of rules were also specialized in the domain of S& T in which we searched for organizations such as universities, faculties, institutes, etc.

NER rules were written using Java Annotation Patterns Engine (JAPE). This engine enables the transformation of a previously tagged text using a grammar based on regular expressions. The grammar consists in a sequence of rules with the form *pattern* → *action* that are executed over the tagged text of each sentence. The left side of the rule specifies a pattern of tags, in the form of a regular expression, and the right side of the rule is an action to be executed if the left side is satisfied by the tagged text. The following rule shows an example of how to identify an organization, in this case a faculty, in Spanish:

```
Rule:OrgFacultadSpanish
Priority: 25
// EXAMPLE: Facultad de Ciencias Exactas
(
    (
        {Token.string == "Facultad"}—
  ({Token.string == "Fac"}({Token.string == "."})?) |
        ({Token.string == "Fc"}({Token.string == "."})?)
    )
    (
        (CONNECTOR_INSTITUCION)?
        ({Token.orth == upperInitial}—{Token.orth ==
  allCaps}|(INITIALS))
        (CONECTOR_INSTITUCION_COMA)?
    )+
):orgName —>:orgName.TempOrganization = {kind =
  "org", rule = "OrgFacultadSpanish"}
```

To match this rule, the text has to begin with the tokens "*Facultad*", "*Fac*", o "*Fc*" (which are possible start tokens for a faculty in Spanish), followed by an optional point, followed by a connector that identifies an institution (*CONNECTOR_INSTITUTION*, defined in another rule), followed by a word that either begins with an uppercase letter or is completely in uppercase or is an acronym (defined in *INITIALS* rule). In the case that a given text matches the left side of this rule, it will be assigned a new tag corresponding to an Organization.

The following rule is an example oh how we identify a Person:

```
Rule: Person Title
Priority: 45
// Example: Sr. Juan García
// Example: Prof. Dr. Juan Perez
(
    (TITLE)+
    (FIRSTNAME)+
    (SURNAME)
    (PERSONENDING)?
)
Macro: TITLE
(
    {Title}
    ({Token.string == "."})?
)
Macro: FIRSTNAME
(
    {FirstPerson.gender == male} |
    {FirstPerson.gender == female} |
    (INITIALS)
)
Macro: INITIALS
(
    (
    {Token.orth == upperInitial, Token.length =="1"}
    ({Token.string == "."})
    )+
)
```

This rule indicate that a string of text will be considered a Person if it starts with one or more titles, followed by one or more names, followed by a surname (the definition of the rule for identifying surnames accept compound surnames) and ending with an optional suffix (such as *Jr.*). This definition depends on other rules that identify titles, names or surnames. The rule for identifying a title, for example, is also shown in the previous code example and consists of a *TITLE* token, which is matched against a dictionary of titles, followed by a dot character.

4.2. Indexing and searching

After the crawling and NER processes, the information retrieval process is done in four steps:

1. Represent documents and Web pages in a suitable form to be indexed. This step consists in obtaining a subset of words that better describe each document and is done during the indexing process.
2. Identify the information needs of the user, represent it in the same form as documents and create a query to the system.
3. Search for documents matching the query by comparing both representations.
4. Score the documents recovered according to their relevance with respect to the query and sort them according to its score.

These features were implemented using Apache Lucene,[10] an open source library for indexing and searching. Details of these processes are given in the following sections.

[10] http://lucene.apache.org/.

4.2.1. Index structure

The process of indexing or creating a searchable index is a basic functionality of any search engine and consists in generating and maintaining the data structures needed to store the content retrieved from a set of Web sites. This process is done after analyzing the text and extracting relevant information from the pages retrieved during the crawling process, in order to optimize future searches for this information.

The result from the previous step consists of a set of pages with a set of associated entities. Using this information as input, indexing is done in such a way that both basic and advanced queries on the indexed text can be efficiently solved.

The index consists basically in an inverted file, a word-oriented mechanism for indexing a collection of texts in order to accelerate the task of searching those documents. In traditional search engines the index contains an entry for each word found in the indexed documents. In turn, each of these entries has a pointer to a list of URLs or documents containing that word. This enables to quickly find a given word in the index and to retrieve the indexed documents that contain that word. SciTechSE index also stores the entities recognized within each page so that they are also recovered in the searching process.

In order to improve the searching process, the inverted file is divided into two parts: a dictionary containing all the words in the index, global statistics of each word (such as term frequencies, inverse document frequencies), and a pointer to a second file, called postings file, which stores information about each of the individual occurrences of a word (including a pointer to the corresponding documents where each word appears).

Thus, the dictionary is used to search for the terms appearing in the query since it only has one entry for each unique term that appears in the indexed information. Once the term is found in the dictionary, the postings file is accessed through a pointer to the place where entries to the corresponding term starts in order to extract more data on each occurrence of that word and the related documents.

Particularly, for indexing and searching we adapted the functionality provided by Lucene, a high performance information retrieval library written in Java. Lucene offers powerful features through a simple API: scalable, high-performance indexing along with powerful, accurate and efficient search algorithms.

4.2.2. Web pages representation

Representation, retrieval and sorting used in SciTechSE is based on the standard Vector Space Model (VSM). In this model documents and queries are represented as n-dimensional vectors, where each dimension corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero and represents the importance of weight of the term in the document.

In other words, each document d_j is represented in the space as a vector of terms:

$$d_j = (w_{1j}, w_{2j}, \ldots, w_{nj})$$

where w_{kj} is the weight of the k^{th} term in document j.

The w_{kj} for a term t_k in a document d_j is obtained considering a local component related to each document and a global component related to the complete collection of indexed documents. The local factor correspond to the frequency of the term in the document tf_{kj}, that is the number of times that t_k occurs in d_j. The global factor is estimated using the inverse document frequency $idf(t_k) = log(N/n_k)$, where N is the total number of indexed documents and n_k is the number of documents that contain the term t_k at least once.

This way, terms are weighted using a function known as $tf - idf$ (term frequency x inverse document frequency) that is computed as the product of both factors:

$$tf - idf(t_k, d_j) = tf_{kj} . log(N/n_k)$$

This weighting scheme formalizes two empirical observations about the text. First, the more times a term occurs in a document, the more important is the term to determine the relevance of the document to a query. Second, the more times the term appears in the collection, the less useful it is for discriminating between different documents.

The index is internally composed of a series of segments. Each segment contains, among other information:

- *Term Dictionary* is a dictionary containing all the terms used in all indexed fields of the entire collection of documents
- The dictionary also stores the number of documents containing a term and pointers to the data frequency and proximity
- *Term Frequency* for each word in the dictionary to store the IDs of all documents that contain the term and frequency of the term in each document
- *Proximity Term*, for each term the position in which the term occurs in each document is stored
- *Term Vectors*, for each field in each document a vector of terms is stored

The raw text extracted from Web pages goes through a series of analyzers before being converted into vectors to be indexed. The first step in this analysis is the tokenization that uses rules to partition the text into tokens. A token is a unit of text, usually a term or word in the document. The following filters or analyzers applied to the text work on the tokens identified.

The value of a field, such as the contents of a document, must be analyzed before it can be indexed. Lucene includes the concept of pipe of analyzers, i.e. different analyzers can be used for indexing and searching. An analyzer examines the raw text and provides the indexable terms that will be used when indexing the document. The simplest analyzer divides the text into tokens separated by white spaces. Other analyzers perform different types of tasks on the tokens they receive from a previously analyzer.

Analyzers used by SciTechSE, in the order in which they are applied, are the following:

1. *WhitespaceTokenizerFactory*: create tokens when finding white spaces (including spaces, tabs, line breaks, etc.). It does not lowercase the text nor removes hyphens.
2. *StopFilterFactory*: removes stop-words, words with little semantic value such as articles (a, an, the), conjunctions (and, or, but), prepositions (in, on), pronouns (that, this) or very common verbs (be, take). This step saves storage resources and unproductive processing time at later stages. SciTechSE uses a standard list of stop-words of the supported languages (Spanish and English) so that this filter removes the words included in this list.
3. *WordDelimiterFilterFactory*: This analyzer joins or separates compound words. For example, it divides words when there is a transition to uppercase or lowercase (WiFi is divided into Wi and Fi), it concatenates series of alphabetic symbols, etc.
4. *ISOLatin1AccentFilterFactory*: This analyzer normalizes accented characters to their unaccented equivalent (it converts é to e, for example), it also normalized umlauts and others.
5. *LowerCaseFilterFactory*: transform all text to lowercase.
6. *RemoveDuplicatesTokenFilterFactory*: it ensures that no duplicate terms appear in the same position. This filter is applied as a last filter since duplicate terms usually appear as a result of a series of filters applied previously.

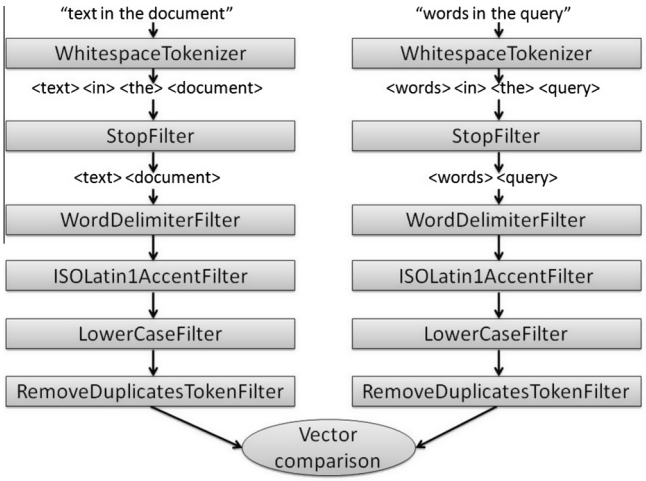

Fig. 3. Analyzers applied to documents and queries.

The same analyzers are applied to the indexed documents and to the queries so that they can be compared in vector space, as shown in Fig. 3.

4.2.3. Queries representation

A query is split into a set of simple terms or phrases. Multiple terms are then combined with Boolean operators such as AND (+), OR and NOT (−) to form more complex queries. The OR operator finds all documents with any of the term in the query (it is equivalent to a union of sets). OR is the default operator, i.e. if there is no operator between two terms is assumed the existence of an OR operator. The AND operator match documents that have both terms expressed in the query (it is equivalent to an intersection of sets). The NOT operator excludes documents containing the term that comes after this operator (it is equivalent to a difference of sets). Finally, SciTechSE allows grouping expressions in parentheses to form sub-queries.

In addition to Boolean operators, the query can be delimited by quotes to find documents containing an exact phrase and it can also contain some special elements, such as:

- ? implies a match of any character. For example, the query "hell? " matches any word starting with "hell" followed by any character.
- ∗ involves multiple matches of any character. For example, the query "hell*" matches any word that begins with "hell" followed by any string.
- ∼ this is a fuzzy operator. It is used at the end of a word, e.g. the query "hello ∼", will return all the words similar to "hello".

The string entered by the user is analyzed as shown in Fig. 3 and then it is taken by Lucene's *QueryParser* module, which translates into a Boolean expression that is then used to find relevant documents. For example, if the interface receives the string "find this text", the parser would transform it into "search OR this OR text".

4.2.4. Searching and retrieval

The retrieval process involves the identification and ranking of documents from the index that are relevant to a given query. SciTechSE combines two classic models of the information retrieval area: the vector space model described in Section 3 and the Boolean model. The Boolean model is used as an initial step to reduce the number of retrieved documents which are then ranked using the criteria of the vector space model.

The Boolean model is a simple model based on the theory of sets and Boolean algebra. The retrieval strategy is based on a binary decision criterion in which a document is relevant or not for a particular query, without considering any degree of relevance (that is, it is a model of exact matching).

The documents retrieved are then those documents that satisfy the logical expression in the query expressed by the user. To achieve this, it uses the operators of Boolean algebra (AND, OR and NOT) and applied them to each term in the index. In the basic search interface provided by SciTechSE the retrieval process returns all documents that contain any of the terms in the query. In the advanced search interface provided by SciTechSE, logic functions can be expressed on the terms of the query.

Once documents are recovered from the index based on this model, SciTechSE uses the vector space model, which is a partial matching model, to sort them according to their degree of relevance with respect to the query. The general idea of this model is that a document is more relevant to a query when the terms of the query appears more often in a document, relative to the number of times that the term appears in all documents in the collection. That is, the weights assigned to the terms are used to calculate the degree of similarity between each retrieved document and the query made by the user.

The traditional measure of similarity in information retrieval systems is the cosine similarity, which measures the angle between the vector of the query q and the vector of each document d_j in the space of vectors. This measure is calculated as the normalized inner product between two vectors:

$$sim(d_j, q) = \frac{d_j \cdot q}{\|d_j\| \|q\|} = \frac{\sum_{k=1}^{n} w_{kj} w_{kq}}{\sqrt{\sum_{k=1}^{n} w_{kj}^2} \sqrt{\sum_{k=1}^{n} w_{kq}^2}}$$

where w_{kj} is the weight of the term t_k in the document d_j and w_{kq} its weight in the query.

The similarity measure used in SciTechSE is inspired in the notions of the cosine measure and is described in the following section.

4.2.5. Scoring and ranking

The scoring function assigns a score to each document in the search results according to their relevance to the query expressed by the user. This score is then used to sort search results before presenting them to the user. All score values are normalized to ensure that all the results have value less or equal to one. The following equation, known as Lucene's Practical Scoring Function McCandless and Gospodnetic, 2010, shows the scoring function used in SciTechSE:

$$score(d, q) = coord(d, q) \cdot queryNorm(q) \cdot \sum_{t \in q} (tf_{td} \cdot idf_t^2 \cdot t.getboost() \cdot norm(t, d))$$

where:

- tf_{td} is the frequency of term t in document d. Documents that have more occurrences of a term that is in the query receive a higher score. This is computed as $\sqrt{frequency}$.
- idf_t is the inverse document frequency, so that rare terms in the collection contribute more to the score. This is computed as $1 + log(\frac{numDocs}{docFreq+1})$.
- $coord(d,q)$ is a factor that considers how many terms in the query are found in the document. A document that contains more query terms gets better score than one that holds only some of them.
- $queryNorm(q)$ is a normalization factor to makes comparable the scores of different queries. This factor does not affect the ranking and all results are multiplied by the same factor. It is computed as: $queryNorm(q) = 1/\sqrt{q.getBoost()^2 \cdot \sum_{t \in q} (idf_{t \cdot t.getBoost()})^2}$

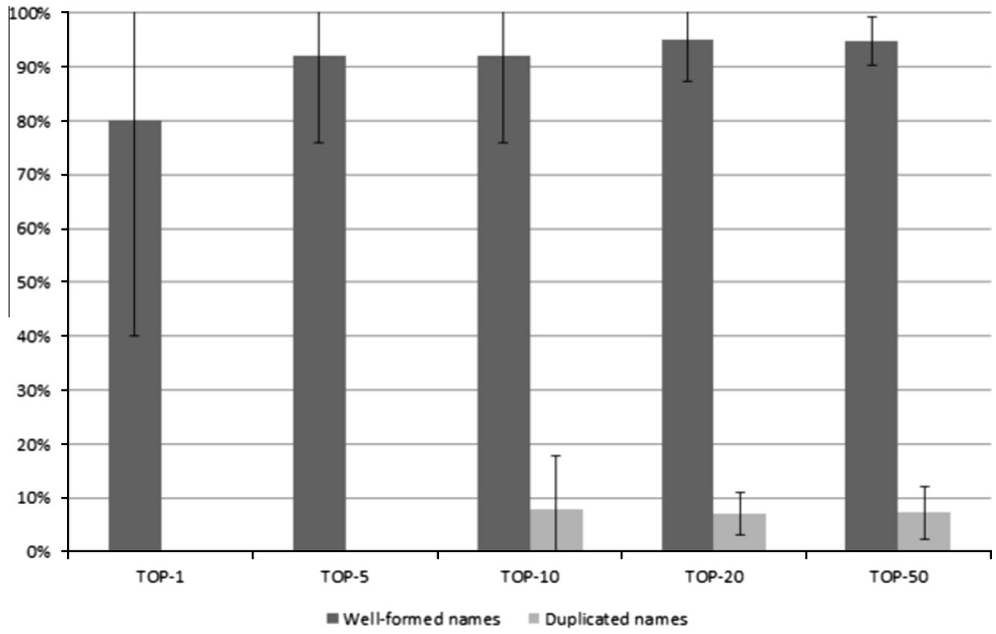

Fig. 4. Average well-formed names and duplicates for five searches.

- *t.getBoost*() is a factor favoring a term *t* in query *q* during the search process.
- *norm*(*t*,*d*) contains factors to boost documents and fields, which are stored during the indexing process, and a normalization factor for length. Namely:
  - Promote a document: a boosting factor assigned to a document before adding it to the index
  - Promote a field: a boosting factor assigned to a field before adding the field to the document
  - *LengthNorm*(*field*) is calculated when the document is added to the index according to the number of tokens in the document field

  These factors are multiplied when adding a document to the index. That is: $norm(t,d) = doc.getBoost() \cdot lengthNorm(field) \cdot \prod_{field\, f \in d\, named\, as} tf.getBoost()$For example, the title of a document is a field that is considered more descriptive than the content of the document. For this reason, a higher boosting value is assigned to this field so that a term of the query appearing in the title of a document gives a better score to that document.

4.3. Improving the user experience

In order to improve user satisfaction, SciTechSE includes a number of additional features: Spell checking, finding similar documents and filtering using facets. We describe next how this features are implemented.

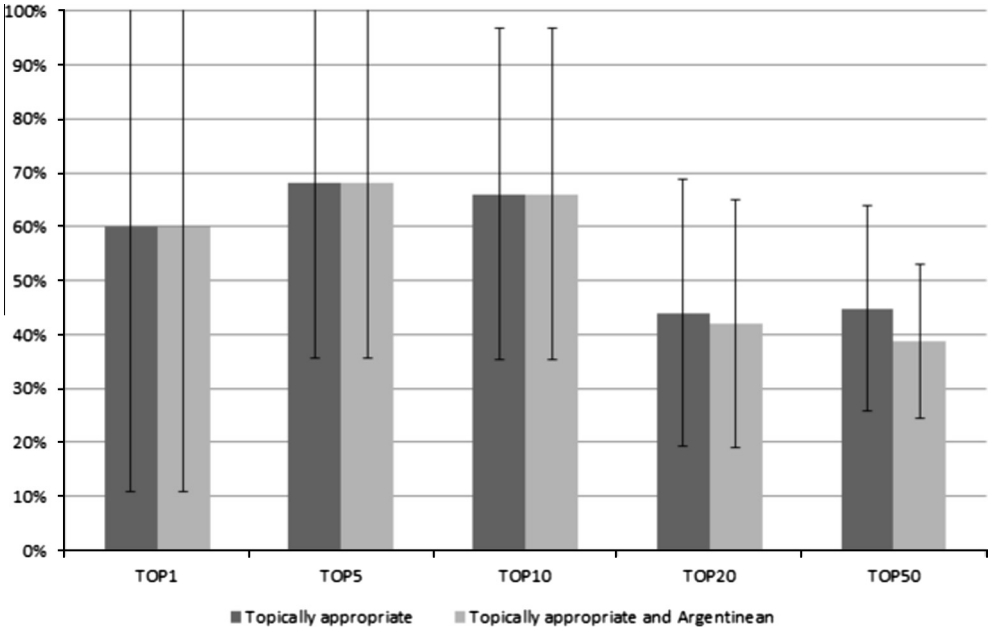

Fig. 5. Average topically appropriate names for five searches.

4.3.1. Spell checking

Most Web search engines correct the spelling of queries suggesting query terms that are similar to the terms used by the user. For example, if the user query is "*artificial intelligence*", the search engine prompts the user with the question "*Did you mean: artificial intelligence?*". To implement this feature, instead of using a dictionary, we use the collection of indexed documents to find terms similar to that in the user query. The set of all terms longer than three characters in the main index are extracted and added to a new n-gram-based *SpellChecker* index (McCandless and Gospodnetic, 2010). The length of the n-grams in this index range from 1–2 for short words and 3–4 for long words. Under the assumption that the characters of the misspelled words are more likely to be found in the inner n-grams, the n-grams of a fixed length are divided into three categories to make it possible to assign higher weights to he n-grams at the beginning or ending of the word. Misspelled words are then split into n-grams using the same procedure and the resulting terms are used to query the SpellChecker index to retrieve the most similar words.

4.3.2. Finding similar documents

Users of SciTechSE can chose to find documents that are similar to any result after performing a query. To implement this feature, a "more like this" link is appended to every search result. When the user clicks on a "more like this" link, the corresponding document is used to create a term vector that will be used to build a new query. Before performing the query stop-words are removed and the resulting terms are sorted by descending order of value. Only the most frequent terms in the document term vector are used to query the index. The number of terms used depends on the length of the document, and corresponds to the square root of the total number of tokens.

4.3.3. Faceted search

The use of faceted browsing is aimed to make it easier to refine search results for both expert and non-expert users. Facets correspond to properties of the information elements that are derived by the analysis of the text of web documents using entity extraction techniques, as detailed in Section 4.1.3.

After performing an initial query, the entities recognized within the search results are grouped into three categories: People, Organization and Places. These facets can then be used to show only results in which a selected entity appears. The process is iterative, so that after selecting an entity and filtering out the matching results, a new set of entities (those appearing in the filtered results) are shown to the user. Conceptually, faceted browsing can be viewed as the Boolean AND of multiple sets representing different attributes.

For example, assume that the user searches for "artificial intelligence" and a set of 100 matching documents are retrieved. Then the user can select "Joe Doe" under the People category and the results will be narrowed to those pages matching "artificial intelligence" which contain Joe Doe as a recognized Person entity. The entities under the People category will no longer include Joe Doe, but all people who co-appear in results containing Joe Doe.

4.4. Front-end for indexing and searching

The browser interface and the crawler do not directly interact with the indexer but through a front-end for indexing and search. SciTechSE uses Sorl[11] for this aim. Solr acts as a server that communicates with both components via HTTP and XML standards acting as a wrapper for Lucene, managing the synchronization of accesses to index and providing multiple levels of caches to resolve queries more effectively. Furthermore, Solr supports the distribution and replication of Lucene indexes so that searches can be distributed if it is necessary to increase the scalability of the system. Solr server runs on Tomcat[12] as a servlet container.

Documents with their various fields and types were defined in an XML schema used by Solr to instruct the indexer about the type of information contained in the documents. Next we show the definition of some of the fields used in SciTechSE, including those containing entities recognized in the text. In this scheme the analyzers that will be applied to each field are also defined.

```
< field name=''host'' type=''url''
  stored=''false'' indexed=''true"/>
< field name=''site'' type=''string''
  stored=''true'' indexed=''true"/>
< field name=''url'' type=''url'' stored=''true''
  indexed=''true'' required=''true"/>
< field name=''content'' type=''text''
  stored=''true'' indexed=''true"/>
< field name=''title'' type=''text''
  stored=''true'' indexed=''true"/>
< field name=''people'' type=''string''
  stored=''true'' indexed=''true''
  multiValued=''true"/>
< field name=''places'' type=''string''
  stored=''true'' indexed=''true''
  multiValued=''true"/>
< field name=''organization'' type=''string''
  stored=''true'' indexed=''true''
  multiValued=''true"/>
```

Documents are added to the index by posting an XML file with the contents of the document through an HTTP POST to Solr. Entities recognized in a document are also added as values of the corresponding field (people, organizations or places). This way, after processing a document with the entity recognizer, an XML file containing all the values for fields in the document is created and posted to the index.

The desired response format is set to the Solr query engine, since not only the documents matching a given query are required, but also *faceted search* and *highlighting*. A set of parameters are sent along with the query, starting with the query string q entered by the user (some previous checks are applied to avoid errors). The language of the retrieved pages is also included in the request.

A query is typically satisfied by a number of indexed documents that are not shown all at the same time but in a paged scheme. This is indicated by the query parameters *start* and *rows*. Other parameters of the queries are defined in Solr's settings file, such as the default operator *q.op* (being the OR operator in our case) or *df*, the default field for searching, (being the content of the document the case for SciTechSE).

Finally, when the user selects any filter in SciTechSE graphical interface they are also added to the same HTTP request that expresses the query. For example, if the user searches using the word "doe", she indicates that she only want pages in Spanish to be retrieved and then she select the person John Doe from the list of entities recognized in the search results, the request will be:

*http://isistan.exa.unicen.edu.ar/buscador/#q = doe;lang = lang_spa;facet = PEOPLE:John%2520Doe;*and the result will be all those pages containing the entity "John Doe".

The result of this request made to Sorl server will be an XML file describing the results in the format specified above. An example is

[11] http://lucene.apache.org/solr/.

[12] http://tomcat.apache.org/.

shown in the following code example, where we can see first the number of documents matching the query (in this case 509), and then the list of the resulting documents. For each document all its data and associated entities of each type are specified.

```
<response>
<responseHeader>...</responseHeader>
<result numFound=''509'' start=''0">
<doc>
<float name=''boost">1.4142135</float>
<str name=''content"> ... text of the page ... </str>
<str name=''lang">en</str>
<arr name=''organization">
<str>Department of Computer Sciences</str>
<str>UNICEN University </str>
</arr> <arr name=''people">
<str>John Doe</str> </arr>
<arr name=''places">
<str>Tandil</str>
</arr>
<str name=''site">www.exa.unicen.edu.ar</str>
<str name=''title"> ... title of the page ... </str>
...
</doc>
...
</result>
</response>
```

The resulting XML files are then processed before being displayed on the search engine interface.

5. Experiments

In order to test the search engine we evaluate two aspects. Firstly, we determined the search engine capacity of extracting valuable entities starting from Web page. Secondly, we determined whether the extracted names correspond to actual researchers. For evaluating both aspects we performed five queries to the search engine and manually classified the results based on our knowledge of the Argentinian research community in the consulted fields of computer science. We used the following queries to judge results for the top 1, 5, 10, 20 and 50 names identified by SciTechSE: "*artificial intelligence*", "*intelligent agents*", "*software engineering*", "*web services*" and "*logic*".

For evaluating the extracting names we considered if the extracted names were well-formatted. This means that corresponds to a name, possibly initials of second names and a surname. Errors included truncated people names (for example, names without a surname) or entities composed of words not corresponding to people names. In this evaluation, we also counted entities that were duplicated in the search results because they have different forms (for example, in one case the name is formed by a surname and a full name and in other case it is formed by the surname and the initials of the name).

Fig. 4 summarize the results of evaluating the previously described issues. The average percentage of well-formed people names and duplicated entities are depicted, error bars shows the standard deviation for the five queries. As it can be observed, the method reaches a very high precision on recognizing people names, whereas a few duplications are included in the list of results.

The evaluation of the results regarding the identification of real researches related to the topic of the query is presented in Fig. 5. First, the percentage of researchers actually working in the area of a query was determined from each query results. We can observe that the number of researchers of the area identified in the top 10 results is high: almost 7 out of 10 results correspond to researchers working in the subject. This number drops considerably if the top-50 results are considered. It is worth noticing that the remaining results belong to researchers that are mentioned in the crawled pages matching the query, but whose research is not completely related to the consulted topic. Finally, we should mention that sometimes Argentinian researchers publish or interact with foreign researchers whose names are also part of the search results for being mentioned in the Web pages (for example, as paper co-authors), we show in the figure the number of Argentinian researchers identified. Since external researchers are mentioned with a low frequency their names only starts to appear after the top-10 results.

6. Conclusions

We have designed a search engine in the domain of science and technology in our country. SciTechSE indexes the institutional sites of universities and scientific and technological institutes of our country and allows users to navigate a large collection of scientific pages using faceted meta-data associated to each indexed document. The facets used correspond to people, institutions and places and they are automatically detected using NLP techniques. Besides faceted search, the presented search engine also provides other common search engine features such as "More-Like-This" document search, highlighting of searched terms, field collapsing of search results and spell-checking of query terms.

This development aimed at closing the gap between academic and productive sectors of the country making more accessible the information about research works and technological advances. The advantage of this search engine is twofold. First, it does not require to upload information into a centralized repository, which would have increase the workload of scientific or institutions, since it process information that they publish on the Web. Second, an intensive text processing approach enables the detection of the main entities involved in the available documents and, in turn, presenting the information in an enriched and friendly manner to non-expert users.

Acknowledgment

This research was supported by the Buenos Aires Scientific Research Commission (CICPBA), Buenos Aires, Argentina.

References

Altingovde, I. S., Ozcan, R., Cetintas, S., Yilmaz, H., & Ulusoy, Ö. (2007). An automatic approach to construct domain-specific web portals. In *Proceedings of the sixteenth ACM conference on information and knowledge management, CIKM '07* (pp. 849–852). New York, NY, USA.

Bar-Ilan, J. (2000). Results of an extensive search for s& t indicators on the web: A content analysis. *Scientometrics, 49*(2), 257–277 [ISSN 0138-9130].

Choi, Y. S. (2011). Tpematcher: A tool for searching in parsed text corpora. *Knowledge-Based Systems, 24*(8), 1139–1150.

Dakka, W., & Ipeirotis, P. G. (2008). Automatic extraction of useful facet hierarchies from text databases. In *Proceedings of the 2008 IEEE 24th international conference on data engineering, ICDE '08* (pp. 466–475). Washington, DC, USA: IEEE Computer Society.

Fellbaum, C. (Ed.). (1998). *WordNet: An electronic lexical database*. MIT Press.

Hatcher, E., McCandless, M., & Gospodnetic, O. (2010). *Lucene in action* (2nd ed.). Manning Publications.

Hearst, M. A. (2006). Clustering versus faceted categories for information exploration. *Communications of the ACM, 49*(4), 59–61 [ISSN 0001-0782].

Hepple, M. (2000). Independence and commitment: Assumptions for rapid training and execution of rule-based pos taggers. In *Proceedings of the 38th annual meeting of the association for computational linguistics (ACL-2000)*.

Hong, J. L., Siew, E.-G., & Egerton, S. (2010). Information extraction for search engines using fast heuristic techniques. *Data Knowledge Engineering, 69*(2), 169–196 [ISSN 0169-023X].

Jacquenet, F., & Largeron, C. (2009). Discovering unexpected documents in corpora. *Knowledge-Based Systems, 22*(6), 421–429 [ISSN 0950-7051].

Patterson, D., Rooney, N., Galushka, M., Dobrynin, V., & Smirnova, E. (2008). Sophia-tcbr: A knowledge discovery framework for textual case-based reasoning. *Knowledge-Based Systems, 21*(5), 404–414.

Prasad, A. R. D., & Guha, N. (2008). Concept naming vs concept categorisation: A faceted approach to semantic annotation. *Online Information Review, 34*(4), 500–510.

Ramirez, P. M., & Mattmann, C. A. (2004). Ace: Improving search engines via automatic concept extraction. In *Proceedings of the 2004 IEEE international conference on information reuse and integration* pp. 229–234.

Sleiman, H. A., & Corchuelo, R. (2013). Tex: An efficient and effective unsupervised web information extractor. *Knowledge-Based Systems, 39*, 109–123.

Tang, T. T., Craswell, N., Hawking, D., Griffiths, K., & Christensen, H. (2006). Quality and relevance of domain-specific search: A case study in mental health. *Information Retrieval, 9*(2), 207–225.

Wong, C.-Y., & Yap, X.-S. (2012). Mapping technological innovations through patent analysis: A case study of foreign multinationals and indigenous firms in china. *Scientometrics, 91*(3), 773–787 [ISSN 0138-9130].

Yee, K.-P., Swearingen, K., Li, K., & Hearst, M. (2003). Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI conference on human factors in computing systems, CHI '03* (pp. 401–408). New York, NY, USA: ACM.