Automatic Gathering of Educational Digital Resources to Populate Repositories

Ana Casali
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario
CIFASIS
Rosario, Argentina.
+54 3414802649 (ext. 141)
acasali@fceia.unr.edu.ar

Claudia Deco
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario
Facultad de Ingeniería y Química.
Universidad Católica Argentina
Rosario, Argentina.
cdeco@uca.edu.ar

Santiago Beltramone
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario
Rosario, Argentina.
santiagobeltramone@gmail.com

ABSTRACT
To populate Institutional Repositories, it is necessary to apply appropriate policies and strategies for dissemination and it is important to develop tools to detect all educational digital objects that are already published on institutional web sites that could be uploaded to a repository. This recopilation is a tedious task and is usually performed manually. In this paper we propose a system architecture for collecting text documents in Spanish or English to assist the manager of institutional repositories in the recopilation task of EDOs within a restricted website. Thus, plausible documents to be uploaded to a repository can be detected. Also, its metadata such as title, category, author, language, keywords and relevant contact data, are automatically extracted in order to ask the author for the publication of his/her document. A prototype of this system was developed and a case study at Universidad Nacional de Rosario (UNR) is analyzed.

Categories and Subject Descriptors
H.3.3 **[Information Search and Retrieval]**: *Information filtering*

General Terms
Management, Documentation, Design.

Palabras Claves
Information Gathering, Educational Digital Object, Repositories Information Extraction.

1. INTRODUCCION
Developing Open Access Institutional Repositories in public universities in Argentina is a priority under the policy of the Ministry of Science, Technology and Innovation. Thus, in recent years the research community in our country has worked on projects to design and transfer a theoretical, methodological and technological experimental model for Educational Digital Object Repository [1].

It is considered that an Educational Digital Object (EDO) is any material in digital format that can be used as an educational resource. For example, a scientific publication, an educational material used in a class or a video is an educational resource.

An open problem in these Institutional Repositories is how to populate with EDOs representing the scientific and academic production of an university. In addition to policies and appropriate dissemination strategies, it is of interest the development of computer tools to detect all documents already published in various web domains of this institution that can be uploaded to the repository. For this, is crucial to get document information such as contact details of the author, since the author is mandatory to authorize the publication of his/her document granting a license. Currently, this task is tedious and this recopilation is performed manually by the repository manager.

Regarding automatic gathering information systems, various proposals have been developed. Agathe [2] is a multi-agent system for information gathering on restricted domains. To restrict the recopilation on the web to a certain domain, the authors use an ontology to consider context information, allowing treat web pages belonging to that domain in a more intelligent way, which implies an improvement in the precision of information extraction. For this purpose, machine learning methods are used with adaptive approaches. The authors of this article applied this system to the recopilation of Call for Papers (CfP). CROSSMARC [3] is a European project of multidomain system based on multilingual agents for extracting information from web pages. It uses an approach based on knowledge combined with machine learning techniques in order to design a robust system for extracting information from websites of interest. Because of the constant change of the web, this hybrid approach supports adaptability to new emerging concepts and a degree of independence from specific web sites considered in the training phase. CiteSeerX [4] is a scientific literature digital library and a search engine which automatically crawls and indexes scientific documents about computer science. Its architecture is based on modular web services and pluggable components. These are excellent reference architectures for gathering information systems and were considered in the developing of this proposal. However, all these works consider documents that have some type of structure such as Call for Papers or scientific papers. Also, CiteSeerX and Agathe consider documents only in English, while our interest is also to collect

INTERACCIÓN 2014, September 10-12, 2014, Puerto de la Cruz

documents that may be in Spanish. Moreover, in all cases previously analyzed, only information that is contained in the document is extracted but they did not explore information that could be in linked websites.

In this paper we propose a system architecture for collecting text documents in Spanish or English to assist the manager of institutional repositories in the recopilation task of EDOs within a restricted website. Thus, plausible documents to be uploaded to a repository can be detected. Also, its metadata such as title, category, author, language, keywords and relevant contact data, are automatically extracted in order to ask the author for the publication of his/her document. Specifically, contact data are email and affiliation of the authors of the document to be uploaded.

A problem that can be found in this extraction is that many times, the required data (author, affiliation and email) are not in the document. These data can be in different pages of the same website. The proposed architecture takes advantage of this feature to improve the automation of information extraction. Therefore, in this system some data or metadata extracted are searched in the document text and are also searched in linked sites. The system receives as input a list of URLs of websites where the search is performed. The output of the system shows the retrieved documents together with the extracted information in a database.

This paper is organized as follows. Preliminary concepts are presented in Section 2 and some extraction tools are analyzed. Section 3 describes the architecture of the system with its main components. In Section 4 shows results of the experimentation. Finally, some conclusions are presented.

2. PRELIMINARIES

2.1 Web Crawling

A Web crawler is a program that inspects web pages in a methodical and automated way [5]. One of its most common uses is to create a copy of all visited web pages for later processing by a search engine that indexes pages providing a fast search. Web crawlers begin visiting a list of URLs, identify the links in these pages and add them to the list of URLs to visit recurrently according to a given set of rules. The usual processing of a crawler is from a group of initial URLs addresses (called seeds) which linked resources are downloaded and analyzed in order to look for links to new resources, typically HTML pages, repeating this process until the final conditions are reached. These conditions vary according to the desired crawling policy.

The behavior of a Web Crawler is a combination of different policies of selection, revisit, diplomacy and parallelization. This work focuses on Academic Focused Web Crawlers. The objective of this type of crawler is to collect academic papers into repositories. Examples of these crawlers are CiteSeerX[1], Google Scholar[2] and Microsoft Academic Search[3]. Basically, to the work of the focused crawler is added the task of selecting specific text formats, such as PDF, PS or DOC. Moreover, they use detection techniques for academic articles in a post-processing, where machine learning algorithms, regular expressions or ad-hoc rules can be employed.

Academic papers are obtained from websites of educational and research institutions. Seed selection plays a major role in the results. However, the full articles that can be found are a minor proportion of the total, as they often are being marketed with restricted rights. The crawler used in our prototype belongs to this category, which also domain restrictions in the URL seeds are applied, since the EDOs of interest are those belonging to our university.

Among open access crawlers are: Heritrix[4], Apache Nutch[5] y Crawler4j[6]. In this work we decided to use Crawler4j for being lightweight, scalable, fast, it is developed in Java and because of its ease of configuration and policies crawling selection, and it has good performance. Remarkably, much larger projects like Nutch and Heritrix, exploit its potential in distributed systems dedicated to permanent crawling tasks, features not required in the prototype presented here.

One of the alternatives evaluated and discarded in this study was the use of queries to external search engines like Google, Yahoo, etc. These engines have crawlers that are permanently working with a wide range of queries in different types of files (PDF, PostScript, etc.). The advantages of queries to them lie naturally in saving processing task. While many information gathering systems use this approach (e.g. Agathe), in the present study could not be applied because it is essential to have the structure of the site which has the link to the EDO of interest. This is because we cannot always find contact data of authors within the document which, as discussed below, can be found in nearby HTML pages.

2.2 Information Extraction, Retrieval and Gathering

The main goal of Information Extraction systems is to locate information from text documents in natural language, producing as output, a structured tabular data without ambiguity, which can be summarized and presented in a uniform way [6]. Increasingly, it is necessary to extract information for different purposes from the web. It can be seen as a large collection of documents written in natural language and distributed on different servers and files of various format. Therefore, the web is a great source for discovering knowledge.

An Information Retrieval system retrieves relevant documents within a larger collection, while an Information Extraction system extracts relevant information in one or more documents. Therefore, both techniques are complementary and used in combination can result in powerful tools for text processing.

Both areas differ in their objectives and use different computational techniques. These differences are due to the inherent objectives and to the history of each area. Much of the work that has emerged in Information Extraction comes from rule-based systems, linguistic computing and natural language processing systems while the field of Information Retrieval is influenced by areas such as information theory, probability and statistics.

The most used metrics to evaluate information retrieval are Precision and Recall. Precision is the proportion of correct answers on the amount of responses. Recall measures the number of correct responses to total possible correct answers. Both metrics take values in the range [0,1] and its optimum is 1.

Because of the great growth of the web and the heterogeneous of its pages, the task of gathering information is increasingly

[1] http://citeseerx.ist.psu.edu/
[2] http://scholar.google.com.ar/
[3] http://academic.research.microsoft.com/
[4] https://webarchive.jira.com/wiki/display/Heritrix/
[5] https://nutch.apache.org/
[6] https://code.google.com/p/crawler4j/

complex. A Gathering Information System is responsible for performing the retrieval and extraction of information in well-defined collections. To retrieve relevant information, the gathering should be restricted to specific domains. Namely, the context in which the information is collected must be considered.

2.3 Tools for Information Extraction in text documents

Until today, there are not many works in automatic extraction of metadata in text documents [7, 8]. Each tool extracts different types of metadata, has its own objectives, architecture and uses different techniques.

In [9] general metadata extraction tools for title, authors, keywords, abstract and language are analyzed. In particular the following tools: KEA[7], MrDlib[8], Alchemy[9] and ParsCit[10] were compared. After conducting various experiments with these tools on a corpus of 760 documents in our university repository, results obtained with respect to the different metadata that can be extracted were analyzed by each of them: MrDLib for Title and Authors, KEA for Keywords and Alchemy for Title, Keywords and Language. It was observed that the results found with KEA and Alchemy on keywords are similar in precision and that the results obtained with MrDLib and Alchemy for title and authors are similar too. Furthermore, Alchemy results were compared with its combination with ParsCit for preprocessing of documents and with the combination Alchemy+ParsCit the best results were obtained. ParsCit allows structuring the document and creates an XML document that attempts to identify Title, Author, Abstract and Keywords. This information is concatenated into a new file, which is used to submit AlChemyAPI server instead of the original file. From the results obtained in this work, it was decided to use in our proposal Alchemy+ParsCit for information extraction and Apache PDFBox[11] to convert plain text file to PDF format. Next, the tools used are briefly described:

AlchemyAPI is a text mining platform which provides a set of tools for semantic analysis using natural language processing techniques. It provides a set of services that allow automatically analyze plain text documents or HTML. This tool includes multiple services from its RESTful API. Among them are: extraction of Author, Entities, Keywords, Content Categorization and Identification of Language. In its free version, the service has a daily limit of 1000 queries and a limit of 150 kbs per query.

ParsCit is an open source application that performs two tasks: parsing reference strings for citation extraction and analysis of the logical structure of scientific papers. These tasks are performed from a text file using supervised machine learning methods using conditional random fields as a learning mechanism. It includes utilities to run as a web service or as a standalone application.

Apache PDFBox is a Java open source code tool for working with PDF documents. It allows creation of new PDF documents, manipulation of existing documents and the ability to extract content from documents. This tool extracts text from these files. In addition, searches for metadata (author, title, organization, keywords, etc.) that can be embedded in the binary file, loaded at the time of the construction of the PDF. The tool parses the Document Catalog of the binary in the content search within the Metadata section.

2.4 Graph Databases

A graph database is a database that uses graph structures with nodes, edges, and properties to represent and store data. A graph database is any storage system that provides index-free adjacency. This means that every element contains a direct pointer to its adjacent elements and no index lookups are necessary. General graph databases that can store any graph are distinct from specialized graph databases such as triplestores and network databases. Graph databases are based on graph theory. They employ nodes, properties, and edges. Nodes represent entities such as people, businesses, accounts, or any other item you might want to keep track of. Properties are pertinent information relate to nodes. Edges connect nodes to nodes or nodes to properties and they represent the relationship between them. Most of the important information is really stored in the edges. Meaningful patterns emerge when one examines the connections and interconnections of nodes, properties and edges.

Graph databases are a powerful tool for graph-like queries, for example, computing the shortest path between two nodes in the graph. This is the reason why we chose a graph database, since the structure of websites can be easily represented by a graph.

In this paper Neo4j was used, which is an open-source graph database. It is a highly scalable open source graph database that supports ACID, has high-availability clustering for enterprise deployments, and comes with a web-based administration tool that includes full transaction support and visual node-link graph explorer. Neo4j is accessible from most programming languages using its built-in REST web API interface and it is one of the most popular graph database in use today.

3. System Architecture

In this section we propose a gathering information system architecture on restricted web domains for uploading institutional repositories with EDOs. Given the characteristics of the domain, the main functional requirements for our system are:

- Collect digital objects in a configurable list of web domains. For this, is planned to establish the domains belonging to national public universities.
- Extract information from authorship, affiliation and contact information (name of the author, university, institution, email).
- Classify documents into categories, such as publications, lecture notes, etc.
- For each relevant object found, store the document in a database along with information obtained for later viewing

The architecture of the proposed system is described considering two conditions that are present in the gathering problem of EDOs. This architecture can be extended to other gathering problems where the following characteristics are observed:

1. Relevance of Documents: all PDF documents found in the seeds URLs are considered relevant. This is particularly true in the domain of interest of EDOs because it is applied to URL seeds of academic web sites.

2. Information distributed on different pages of the same web site: we start from the empirical observation that the information about an author (name, affiliation, email) are often not in the

[7] www.nzdl.org/Kea/index_old.html
[8] www.mr-dlib.org
[9] www.alchemyapi.com
[10] wing.comp.nus.edu.sg/parsCit/
[11] http://pdfbox.apache.org/

same document or on the website referred to that document but it could be found on another page of the same website. This is very common in educational resources pages. For example, the resource may be in one page, while the contact information of the teachers (possible authors) is in another. As well as on the websites of researchers, where information related to their publications cannot contain contact information, which generally is in the root of their website.

The proposed architecture differs from previous architectures [2, 3, 4] because it deals with documents in Spanish and because of the representation in a graph database of the web sites structure related to seed URLs. This allows extracting information not only on the page of interest but also of related pages.

To achieve the required functionality we designed a modular system with a centralized architecture, where the flow of tasks and information exchange is handled by a coordinator component. The proposed architecture is shown in Figure 1 and its main components are described below.

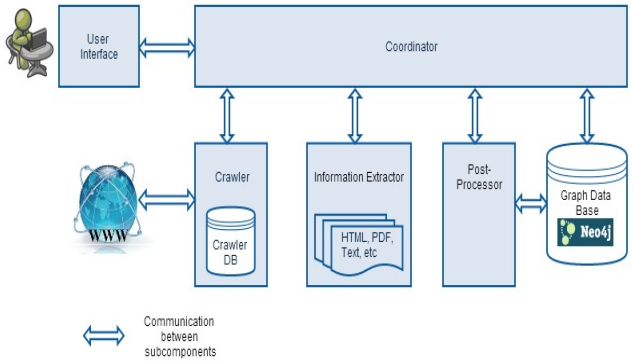

Figure 1. Proposed Architecture

User Interface: is the means by which the user (contents administrator of a repository) initiates and configures gathering information tasks and can view the resulting data.

Coordinator: is responsible for coordinating and communicating the remaining components. Its purpose is to maintain modules isolated from other components so that, they can be easily modified. It manages processes of crawler initialization, persistence in the graph database, post-processing at the end of the crawling and data visualization by the user.

Crawler: this module performs the specific tasks of web crawling. When a resource, which can be in different formats (HTML, PDF, CSS, etc.), is targeted and based on the selected policy of crawling, the crawler notifies the coordinator of the resource found. The coordinator module is responsible for requesting information extraction of that resource and then redirect along with the extracted information to the graphic database for persistence.

Information Extractor: This component is responsible for performing the extraction of information from different resources retrieved by the crawler. It consists of several specialized modules. Depending on the application domain, it may be necessary to implement this component with sub-modules that specialize according to the topic of information to extract or the file type. For domains with different types of documents (e.g. papers, educational resources, video, etc.) and presented in different formats (e.g. PDF, HTML pages, etc.) it is beneficial to use an architecture that have different specialized modules in the extraction of information from each particular topic. In this work different files are processed with different tools.

PDFBox extracts text from a PDF file. Also, this tool searches for metadata (author, title, keywords, etc.) that can be embedded in the binary file and were loaded by the author at the time of the construction of the PDF.

ParsCit performs structure analysis of scientific documents and recognizes different sections of an EDO: title, authors, emails, affiliations, abstract, among others. AlchemyAPI is used for the extraction of keywords and recognition of entities through web services, which receives as input HTML and returns output in XML or JSON format. Specifically, the entities we are looking for are of type organizations and individuals, as potential values of filiation and author fields, respectively.

Graph Data Base: persists the structure of the websites that have been crawled. In the database is generated a graph where nodes correspond to URLs retrieved by the crawler and the leaves are either URLs that have no other outgoing URLs or resources of a certain type of format that are the system target. In Figure 2 we show the database resulting from crawling a site of a researcher professor at UNR.

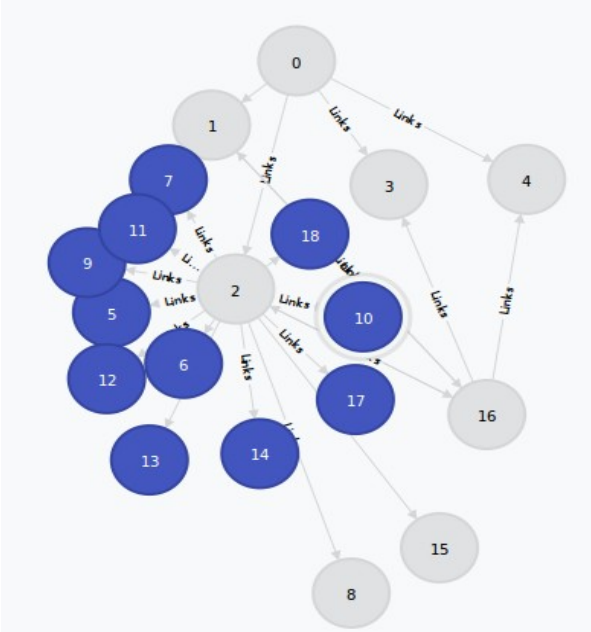

Figure 2. Graph Data Base resulting from crawling a researcher professor website

The gray nodes correspond to HTML pages, while blue ones represent files in PDF format. For each node, the database stores information extracted by the Extractor and the Crawler (father URL, sons URLs, domain, size). Also, it provides to the Post-Processing component routes between nodes in the tree.

Post-Processor: once completed the process of crawling, the coordinator is notified and initiates the Post-Processing. In it, the retrieved nodes are visited and from each one, the process goes bottom-up in the hierarchy structure of the web site that was persisted in the Graph Data Base. The graph path starts in the leaves nodes and continues to a certain distance depending on the system configuration. The aim is to find and link extra information that could not be found on the leaf node and it is

expected to be found in a not so distant node. For example, for the proposed prototype, we look for a contact email and possible information related to authors and affiliations.

3.1 Processing Sequence and Interaction between Components

Figure 3 shows the sequence of processing and communication between the system components.

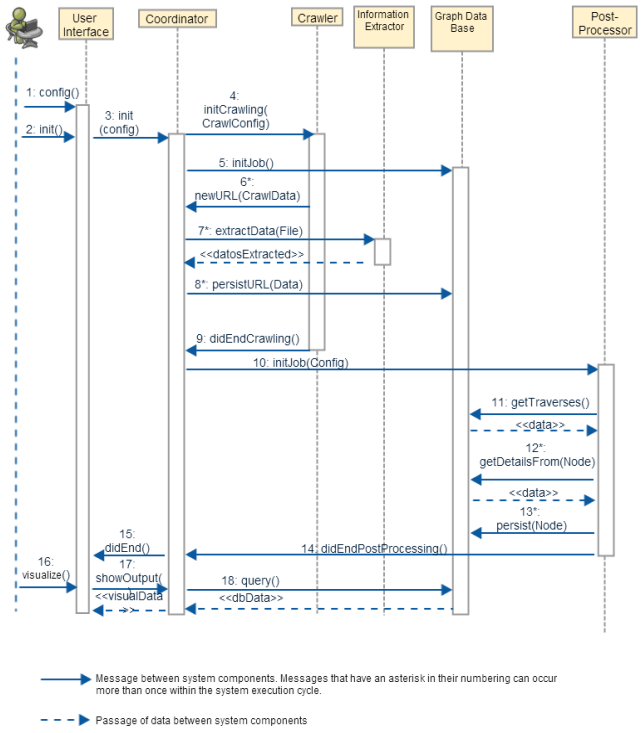

Figure 3. Sequence and Interaction between System Components.

The system administrator is in charge of configuring and starting the processing through the system Interface (Messages 1 and 2). This interface informs the Coordinator (Message 3), who is responsible for the various stages of processing, interacting with and configurating the remaining components. The second interaction is from the Coordinator to the Crawler where the first notifies the second the starting of a new job of crawling (Message 4). For each new file found from a URL the Crawler notifies the Coordinator, who interacts with the Extractor to integrate the data extracted with the data from the Crawler to persist the information in the Graph database (Messages 6, 7 and 8). When the crawling is finished, the Coordinator is notified and communicates with the Post-Processor (Messages 9 and 10). The latter, requests path information from the graph database structure of the seed websites and trace them according to the set configuration (Message 11). It also asks for this information for each node in the path and establishes new associations with information that can potentially be linked (Messages 12 and 13). When the work is finished, it notifies the Coordinator who in turn submits it to the Interface (Messages 14 and 15). Finally, the user can request data visualization, resulting from queries that the Coordinator submit to the Database and the results obtained can be displayed through the interface (Messages 16, 17 and 18).

In Figure 4 we show the output of the resulting extracted data in database.

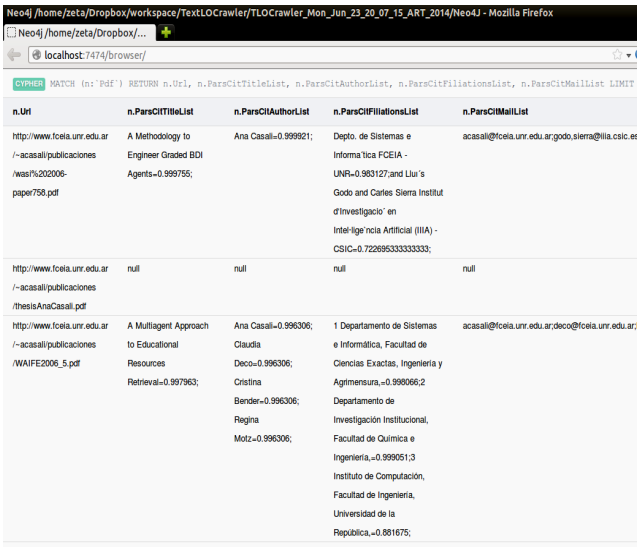

Figure 4. Results Data

We have developed a first prototype of the proposed gathering system. It was implemented in Java using the implementation in the same language of the web crawler Crawler4j and the graph database engine Neo4j as embedded database. The web client of the graph database is also used for data visualization. The wrappers of information extraction tools: Apache PDFBox, Alchemy API and ParsCit were also implemented in Java.

4. EXPERIMENTATION

To evaluate the performance of the proposed prototype we run it in the institutional site of an UNR researcher professor. Our objective was to analyze the feasibility of the proposed architecture and to evaluate the results obtained by different extractors in order to design the system modules with the combination that provides the best results.

From the initial URL, following the steps above described, all the PDF files were retrieved by the Crawler and the database to persist the analyzed site structure was generated. Then, using PDFBox, these files were obtained in text format. In order to analyze which is the best combination of the selected extractors, different experiments have been conducted and some of them are detailed below.

In a first step, we executed the information extraction using ParsCit. As output we got a XML file from which it was obtained the field values: Title, Authors, Affiliations and Emails. For example, we show in Table 1 the analysis of the results obtained for the first 10 documents gathered in this site.

In a second stage, to the text files obtained by the PDFBox was applied a combination of extractors and search using the graph database. This step is performed by the Post-Processor module of the architecture.

To extract the emails first, the system searched regular expressions corresponding to emails in the first page of the document and then, these expressions were sought in the HTML of the family links hierarchy.

Table 1. Extraction results using ParsCit

Document	Title	Authors	Filiation	Email
Doc1	T	PE	T	NA
Doc2	T	PI	T	T
Doc3	T	T	T	T
Doc4	T	T	PI	T
Doc5	T	T	PI	T
Doc6	T	E	E	E
Doc7	T	T	PI	NA
Doc8	T	PI	PE	NA
Doc9	T	T	T	NA
Doc10	T	T	T	E

Where the nomenclature used represents:

T: the complete or total information is retrieved (precision = 1 and recall = 1);

PE: correct, complete or partial information is retrieved, but also other extras erroneous data are extracted (precision < 1 and recall ≈ 1);

PI: partial incomplete, the correct value is not fully retrieved but no erroneous data is obtained (precision = 1 and recall < 1);

E: incorrect;

NA: not applicable, the data was not present where it was looked for;

T (n): the data is correctly found in the inheritance level n.

For authors and their affiliations, the system retrieved the HTML of the linked upper nodes (up to a configurable level) and they were sent to Alchemy-API to extract language and entities of person and organization types, which are respectively associated with potential authors and affiliations.

Regarding the language, 100% correct data is obtained. The analysis of the other results of this process is shown in Table 2 (Authors, Affiliations and Mail columns). To refine the results for Authors and Affiliations, since correct values were obtained with the addition of erroneous data (PE), a filtering process of the values obtained was performed, retrieving only those found on the first page of the analyzed document. With this filter, improvements were obtained in the extraction of this information as can be seen in columns Filtering Author and Filtering Affiliation in Table 2.

Table 2. Results obtained by Post-Processing 1: Extraction in family nodes

Doc	Author	Filiation	Mail	Filt-Author	Filt-Filiation
Doc1	PE	PE	T(2)	PI	-
Doc2	PE	PE	T(2)	PI	PI
Doc3	PE	PE	T(2)	-	PI
Doc4	PE	PE	T(2)	PI	PI
Doc5	PE	PE	T(2)	PI	PI
Doc6	PE	PE	T(2)	PI	T
Doc7	PE	PE	T(2)	PI	PI
Doc8	PE	PE	T(2)	PI	-
Doc9	PE	PE	T(2)	PI	-
Doc10	PE	PE	T(2)	PI	-

In a last step, an additional processing (Post-Processing 2) was performed to combine the extraction results obtained for Authors and Filiations fields resulting from ParsCit and the combination of extraction processes described in the above step (Post-Processing 1). Then, it was considered the union of the data obtained by these two processes for each field and the analysis of the results is shown in Table 3.

Table 3 Extraction Results for Authors and Affiliations using Post-Processing 2

Document	Lang.	Title	Mail	Author	Filiation
Doc1	T	T	T	T	T
Doc2	T	T	T	PI	T
Doc3	T	T	T	T	T
Doc4	T	T	T	T	PI
Doc5	T	T	T	T	PI
Doc6	T	T	T	PI	T
Doc7	T	T	T	T	PI
Doc8	T	T	T	PI	PE
Doc9	T	T	T	T	T
Doc10	T	T	T	T	T

Results of this experimentation are shown in Table 4. We can observe that in the first stage using ParsCit we retrieve 100% of the titles correctly. The authors are correctly extracted in 60% and partially (PE + PI) 30%. Regarding affiliations we get 50% of correct information, 30% partially incomplete and 10% PE. Regarding the extraction of email contact, only 40% correct was extracted and it is noteworthy that in 40% it was not available (NA) within the document. Using Alchemy it was retrieved 100% of the language and applying the combination of extractors (Table 2) for both the author and affiliation field, it was extracted correct information, usually with maximum recall but also, additional incorrect information (PE) was retrieved. After applying the filter, in most cases the information obtained decreased, resulting partially correct 90% for authors and 50% for affiliations. In the latter case, 10% of total information is added. Finally, combining the results obtained by these two forms of extraction we were able to extract correctly 100% of language, title and email contact, 70% of authors and 60% of affiliations with total recall, and the remainder of these fields values partially. Then, we can conclude that this post-processing provides the best results.

Table 4. Extractors comparison

	Lang.	Title	Mail	Author	Filiation
ParsCit	-	100% T	40% T 40% NA	60% T 30% PE+PI	50% T 40% PE+PI
Alchemy	100% T	-	-	-	-
Post-Processing 1			100% T	90% PI	10% T 50% PI
Post-Processing 2	100% T	100% T	100% T	70% T	60% T

5. CONCLUSIONS

In this paper we have proposed an architecture for automating the task of text document gathering within a restricted web domain, in order to detect educational digital objects plausible to be loaded in an institutional repository. We have implemented a prototype that allowed us to evaluate the feasibility of the proposed architecture and experiment the combination of extraction tools to achieve the best results in the fields we are interested. From the analysis of the results obtained, it was decided to design an extractor that combines extraction tools such as ParsCit, Alchemy and the use of regular expressions, posed as a Post-Processing that take advantage of the graph database of the site. With this Post-Processing it was extracted in 100 % correct values for language, title and email contact, 70% of authors and 60% of affiliation and the remainder of these fields' values partially.

The main differences of this proposal with respect to other gathering information systems is on the one hand, the incorporation of graph databases to represent the structure of the web sites related to seed URLs allowing selection and extract information from families linked nodes. On the other hand, we treat with the gathering of diversity types of documents and we extract information in both English and Spanish. It is expected that this tool will be very useful to institutional repositories administrators since it automates an important part of the task of collecting documents.

6. ACKNOWLEDGMENTS

This work was partially supported by Red CYTED RIURE: Red Iberoamericana para la Usabilidad de Repositorios Educativos, by Redes Internacionales VII: Red para la Integración de Universidades en el uso de TICs para la Inclusión en la Educación Superior, and by the LATIn Project: Latin American open Textbook Initiative, Alfa III DCI-ALA/19.09. 01/11/21526/279-155/ALFA III (2011)-52.

7. REFERENCES

[1] San Martín, P., Bongiovani, P., Casali, A., Deco, C.. Socio-technological perspectives for Open Access Repositories development in the context of public universities in the central-eastern Argentina. PKP Scholarly Publishing Conference, DF Mexico. 2013.

[2] Espinasse B., Fournier S., Albitar S., Combining Agents and Wrapper Induction for Information Gathering on Restricted Web Domains, Research Challenges in Information Science, Nice, France, 2010.

[3] Pazienza, M.I., A. Stellato, and M. Vindigni, Combining Ontological Knowledge and Wrapper Induction Techniques into an e-Retail System, in Workshop on ATEM03 held with ECMLlPKDD 2003, Cavtat. 2003.

[4] Huajing Li , Councill I. , Bolelli L. , Ding Zhou , Yang Song, Wang-chien Lee An , Sivasubramaniam C. Lee Giles. CiteSeer X- A Scalable Autonomous Scientific Digital Library, Department of Computer Science and Engineering, The School of Information Sciences and Technology, Pennsylvania State University, 2007.

[5] Castillo, C. PhD Thesis: Web Crawling, Dept. of Computer Science - University of Chile November, 2004.

[6] Eikvil L. Information Extraction from WWW, Norwegian Computer Center, Oslo, 1999.

[7] Pire T., Espinase B. Casali A. Deco C. Extracción automática de metadatos de objetos de aprendizaje: un estudio comparativo, 2011.

[8] Casali A., Deco C., Romano A., Tomé G. An assistant for loading Learning Object Metadata: An ontology based approach. Interdisciplinary Journal of E-Learning and Learning Objects, IJELLO, Volume 9, pages 77-87. Informing Science + IT Education, 2013.

[9] Casali, A., Deco, C., Bender, C., Fontanarrosa, S., Sabater, C. Asistente para el Depósito de Objetos en Repositorios con Extracción Automática de Metadatos. XV Simposio Internacional de Tecnologías de la Información y las Comunicaciones en la Educación (SINTICE 2013), pp 133-136. Madrid, España, 2013.