An Assistant to Populate Repositories: Gathering Educational Digital Objects and Metadata Extraction

Ana Casali, Claudia Deco, and Santiago Beltramone

***Abstract*— This paper presents an assistant to populate institutional repositories. This tool can detect all educational digital objects in a text format that are already published on institutional Websites and can 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 automating this task of collecting text documents within a restricted domain in order to detect plausible documents that can be loaded into a repository. In addition, its metadata, such as language, category, title, authors, and their contact data, is automatically extracted. A prototype of this system was developed, and case studies in two different domains are analyzed.**

***Index Terms*— Information gathering, educational digital objetcs, repositories, metadata extraction.**

I. Introduction

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 a 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, it 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 the treatment of 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 multi domain 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 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

Manuscript received December 9, 2015; revised March 16, 2016; accepted March 16, 2016. Date of publication May 20, 2016; date of current version June 1, 2016. This work was supported by the Programa de Red Ciencia y Tecnología para el Desarrollo through the Project Red Iberoamericana para la Usabilidad de Repositorios Educativos under Grant 513RT0471.

A. Casali is with the Computer Science Department, Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario (UNR), Rosario S2000BTP, Argentina, and also with the Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas, UNR, Rosario 2000, Argentina (e-mail: acasali@fceia.unr.edu.ar).

C. Deco is with the Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario, Rosario S2000BTP, Argentina, and also with the Universidad Católica Argentina Sede Rosario, Rosario S2002QEO, Argentina (e-mail: deco@fceia.unr.edu.ar).

S. Beltramone is with the Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario, Rosario S2000BTP, Argentina,
(e-mail: santiagobeltramone@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/RITA.2016.2554018

1932-8540 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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 II and some extraction tools are analyzed. Section III describes the architecture of the system with its main components. In Section IV, we present a case study to evaluate the proposed architecture, to compare extractors and to decide which the best option is. Finally, some conclusions are presented.

II. Preliminaries

A. 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) where 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 are often being marketed with restricted rights. The crawler used in our prototype belongs to this category, where also domain restrictions in the URL seeds are applied, since the EDOs of interest are those belonging to our university.

In this work we decided to use Crawler4j (code.google.com/p/crawler4j/) for its being lightweight, scalable, fast, and developed in Java, and because of its ease of configuration and policies crawling selection, and its good performance. 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 it 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.

B. 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.

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

The most used metrics to evaluate information retrieval are Precision and Recall. Precision is the ratio of the relevant

[1] http://citeseerx.ist.psu.edu/
[2] http://scholar.google.com.ar/
[3] http://academic.research.microsoft.com/

answers retrieved to the total number of responses. Recall is the ratio of the number of correct answers retrieved to the total number of correct answers. Both metrics take values in the range [0, 1] and its optimum is 1.

These same metrics with the necessary adaptations are used in information extraction. Then, Precision (P) and Recall (R) are defined as follows:

P = # Relevant Items Retrieved/# Items Retrieved

R = # Relevant Items Retrieved/# Relevant Items in the Collection

To evaluate the performance of an information extraction system the two metrics need to be considered.

C. Tools for Information Extraction in Text Documents

In [9] general metadata extraction tools for title, authors, keywords, abstract and language are analyzed. In particular the tools KEA,[4] MrDlib,[5] Alchemy[6] and ParsCit[7] 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[8] 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 automatically allow analyzing 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.

[4]www.nzdl.org/Kea/index_old.html

[5]www.mr-dlib.org

[6]www.alchemyapi.com

[7]wing.comp.nus.edu.sg/parsCit/

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

Apache PDFBox is a Java open source code tool for working with PDF documents. It allows the 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.

Matching Techniques of Proper Names: The different variants or formats in which an author's name or his/her filiation may be written make it very difficult to apply an exact match. A common problem is that different entities extractors for the same text document can extract different strings for the same entity. For example, the strings "Gloria Perez", "G. Perez" and "Perez, G." represent the same person in different formats. Different approximate matching techniques based on phonetic coding or pattern matching have been proposed in [10]. The Levenshtein distance [11] between two strings is defined as the minimum number of editing operations (insertion, deletion or substitution) to convert a string into the other. The Damerau-Levenshtein distance is a variation of the Levenshtein one considering the transposition of a character as another basic operation of cost 1. The Smith-Waterman algorithm [12] is one well-known strategy originally proposed for optimal local alignment of biological sequences determining similar regions between two sequences. It is a similar to the edit distance, with the addition that it allows hops or spaces that mismatch between the chains. In this paper, the similarity measure of the Smith-Waterman algorithm with optimal local alignments was used to search for entities (author and affiliation) in snippets and for the deduplication of authors. The similarity measure Damerau-Levenshtein algorithm was used in the deduplication of institutions such as affiliation because it works better in longer chain lengths and it does not seek a partial alignment than the Smith-Waterman algorithm.

D. Graph Databases

A graph database is a database that uses graph structures with nodes, edges and properties to represent and store data so that graph theory can be applied to explore the database [13]. 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. 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. For these domains, BDOG are an attractive option because they are more efficient for insertion, storage and query in these graphs with respect to other models such as RDF triples.

In this paper Neo4j (www.neo4j.org/) was used, which is an open-source graph database implemented in Java. 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 databases in use today. This engine has graph traversal algorithms that allow saving programming time and is optimized for managing such data.

III. 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:

- Collecting digital objects in a configurable list of web domains. For this, it is planned to establish the domains belonging to national public universities.
- Extracting information from authorship, affiliation and contact information (name of the author, university, institution, email).
- Classifying documents into categories, such as publications, lecture notes, etc.
- For each relevant object found, storing 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 seed 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. Other document formats could be collected, setting them in the Crawler and replacing in the extractor module, PDFBox by another tool to extract text from documents (e.g., Apache Tika http://tika.apache.org/1.4/formats.html).

2. Information distributed on different pages of the same website: we start from the empirical observation that the information about an author (name, affiliation, email) are often not in the 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. The same applies to the websites of researchers, where information related to their publications cannot contain contact information, which is generally in the root of their website.

The proposed architecture differs from previous architectures [2]–[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.

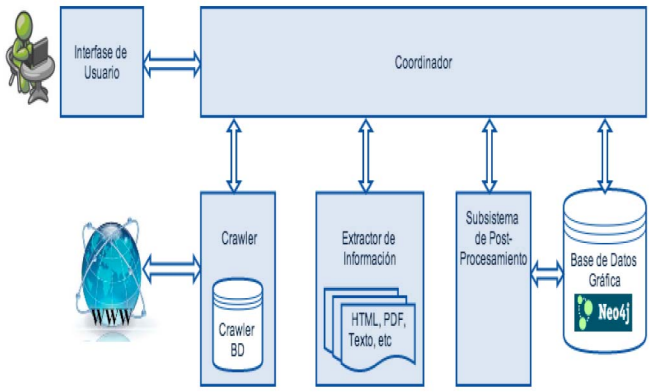

Fig. 1. Proposed architecture.

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 has 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.

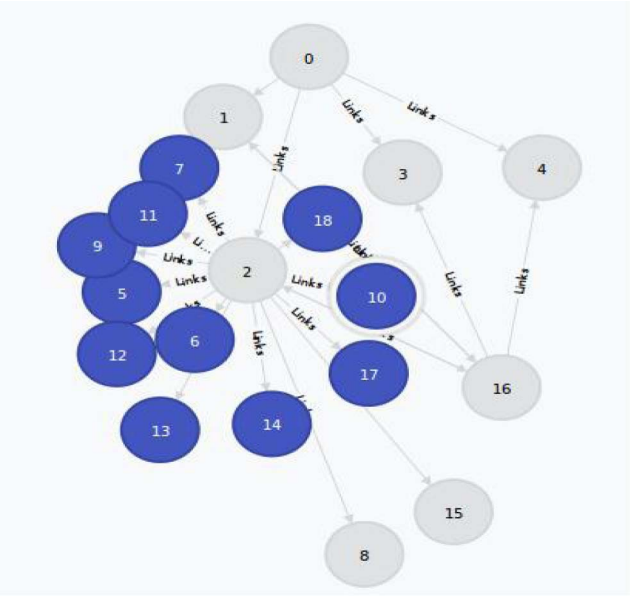

Fig. 2. Graph data base resulting from crawling a researcher professor website.

Graph Data Base: Keeps 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.

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 (parent URL, child URLs, domain and 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 website that was persisted in the Graph Data Base. The graph path starts in the leaf 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.

Two types of post-processing were implemented: EI2 and EI3.

EI2 search *web pages* in an ascendant way in the hierarchy of links searching entities author or affiliation (using AlchemyAPI). All found entities are then filtered, leaving only those that appear in the first few characters of the document. The aim of this extractor is to find entities within the document that could not be recognized by extractors which only analyze the document text.

EI3 searches the existence of such entities or author's affiliation in *documents* found in web pages linked to that node. These entities are then filtered in the first characters of the analyzed document. Due to the different structures of documents, an entity that is in more than one document can be only recognized in some of them. Then, as related documents usually share some of the authors (e.g., lecture notes or publications) the same entity author or affiliation are likely to be present in more than one EDO.

In both extractors, the search of entities is performed with the approximate matching techniques discussed above. At the end of the object post-processing, the union of all the values obtained from different extractors is computed and duplicates are filtered.

A first prototype of the proposed gathering system was developed. Its objective is to obtain a database with the metadata of interest in the documents collected. Then, the repository administrator can decide what EDOs have authors of the institution in order to invite them to upload these resources in the repository. Negative cases may be collected as a resume (which is not considered an EDO) or a scientific article written by external authors to be used as class material (these ODEs cannot be uploaded in the local repository). These cases can be easily detected by the administrator from the extracted information.

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

IV. Case Study

To evaluate the performance of the proposed prototype we ran it in institutional sites at the National University of Rosario (UNR), Argentina. In particular, we chose as case studies two representative sites, one of a teacher-researcher and another corresponding to a subject of the Computer Science Degree. These two types of sites are more limited compared with Department ones and then, the evaluation of extraction results is simpler to analyze but they are sufficiently general because institutional sites contain pages of both types.

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. We focused the analysis on the extraction of title, authors, affiliations, language, e-mail address and category of the document.

From the initial URL, following the steps described above, 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 were conducted and some of them are detailed below. In a first step, we executed the extraction of Language, Title, Authors and Affiliations using ParsCit. Then, the extraction of Authors and Affiliations was refined using EI1 and EI2 Post-Processings. Finally, we considered the union of results from the different extractors applied.

TABLE I
PRECISION AND RECALL USING ParsCit

Doc	Title		Authors		Affiliations	
	P	R	P	R	P	R
1	1.00	1.00	1.00	0.33	1.00	1.00
2	1.00	1.00	1.00	0.33	0.50	0.33
3	1.00	1.00	1.00	1.00	1.00	1.00
4	1.00	1.00	1.00	1.00	1.00	0.50
5	1.00	1.00	1.00	1.00	1.00	1.00
6	1.00	1.00	0.00	0.00	0.00	0.00
7	1.00	1.00	1.00	1.00	1.00	0.50
8	1.00	1.00	1.00	0.33	1.00	1.00
9	1.00	1.00	1.00	1.00	1.00	1.00
10	1.00	1.00	1.00	1.00	1.00	1.00
Avg	1.00	1.00	0.90	0.70	0.85	0.73

TABLE II
PRECISION AND RECALL USING EI2 AND EI3

	EI2				EI3			
	Authors		Affiliations		Authors		Affiliations	
	P	R	P	R	P	R	P	R
1	1.00	0.66	NA	0.00	1.00	0.66	1.00	1.00
2	1.00	0.66	1.00	0.50	1.00	0.66	1.00	0.50
3	1.00	0.50	1.00	0.33	1.00	1.00	1.00	0.33
4	1.00	0.40	1.00	0.50	1.00	0.20	1.00	0.50
5	1.00	0.66	1.00	0.50	1.00	0.33	1.00	1.00
6	0.83	0.83	1.00	0.50	1.00	0.16	1.00	0.33
7	1.00	0.66	1.00	0.50	1.00	0.33	1.00	0.33
8	1.00	0.66	NA	0.00	1.00	0.66	1.00	0.33
9	1.00	0.66	NA	0.00	1.00	0.66	1.00	1.00
10	1.00	0.66	NA	0.00	1.00	0.66	1.00	1.00
	0.98	0.64	1.00	0.28	1.00	0.54	1.00	0.63

TABLE III
PRECISION AND RECALL USING THE UNION OF EXTRACTORS ParsCit, EI2 AND EI3

Doc	Authors		Affiliations	
	P	R	P	R
1	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00
3	1.00	1.00	1.00	1.00
4	1.00	1.00	1.00	1.00
5	1.00	1.00	1.00	1.00
6	0.83	0.83	0.50	0.50
7	1.00	1.00	1.00	1.00
8	1.00	0.67	1.00	1.00
9	1.00	1.00	1.00	1.00
10	1.00	1.00	1.00	1.00
Avg	0.98	0.95	0.95	0.95

To evaluate the system performance, the results obtained at the two sites were analyzed. Precision (P) and Recall (R) of results were computed. In both indicators, conventions for border cases were applied. If the extractor does not produce answers (null denominator in P), we represent this case with the value P=NA (not applicable), since we cannot measure Precision in this case. If there are no correct answers (null denominator in R) and the extractor returns an incorrect result, we set P=0 and R=0. If the extractor does not return any results and in order to highlight the lack of correct answers, it is represented with value R=E (empty). In this way, we can distinguish the overall effectiveness of an extractor by measuring the Recall over total data to be extracted and penalizing with the Precision the generation of spurious data.

Case Study 1: We analyze the site of a teacher-researcher at the University. On this site, the system retrieved all PDF documents that were in it, without bringing unwanted documents from other sites. The analysis of the results obtained for the first 10 documents gathered in this site is discussed below.

The language detection by the extractor Alchemy was correct on all processed documents, why the measures of P and R are 1 for both and are not included in the following table. Concerning the classification of Category (Publication, Thesis, Class Material, etc.) in 90% of cases the documents were correctly labeled with good precision and maximum recall.

In Table I, results of P and R of the ParsCit extractor for Title, Authors and Affiliations are shown. For Authors P and R, they have an average of 90% and 70% respectively. Regarding Affiliation, we have an 85% in P and 73% in R. In the case of Title extraction is optimal, 100% in both measures.

In Table II, P and R for Authors and Affiliations extracted using EI2 and EI3 are shown.

We observe here excellent precision but moderate levels of recall. The intersection of results of the extractors ParsCit, EI2 and EI3 is not empty in most cases. EI2 achieved precisions of 98% and 100% and recall of 64% and 28% for authors and affiliations respectively. EI3 achieved 100% in both entities and a recall of 53% and 63% respectively.

Table III shows Precision and Recall using the Union of Extractors ParsCit, EI2 and EI3. It can be observed that Recall for Author and Affiliation improves over each extractor alone without losing Precision. That is, with nearly optimal Precision (98% and 95%) Recall increased by 25% in the case of Authors and 22% for Affiliations. This indicates that the incorporation of data found on the website where the documents were retrieved improves Recall without losing Precision.

For email extraction three processings were used: *MailList*, which retrieves mails in the first page of the document using regular expressions; *ParsCitMailList*, which retrieves mails using ParsCit; and *RelMailList*, which retrieves mails in related nodes. Then, *UnionMailList* takes the union of the lists above, deleting duplicates. In Table IV, results of extracting mails are shown. We can observe that even the Recall is low for ParsCitMailList and MailList most of the EDOs have at least one email for contacting the authors. As in the case of Authors and Affiliations, by calculating the union of the results it is possible to improve Precision and Recall, reaching optimum in P. Although R is not optimal, the Union collected the email of 60% of the authors, noting that at least one contact mail has been retrieved for each of the documents.

TABLE IV
PRECISION AND RECALL FOR EMAILS EXTRACTION

	MailList		ParsCitMail List		RelMailList		UnionMailList	
	P	R	P	R	P	R	P	R
1	NA	E	NA	E	1.00	0.33	1.00	0.33
2	NA	E	NA	E	1.00	0.33	1.00	0.33
3	1.00	0.67	1.00	1.00	1.00	0.33	1.00	1.00
4	1.00	0.50	1.00	1.00	1.00	0.25	1.00	1.00
5	NA	V	1.00	1.00	1.00	0.25	1.00	1.00
6	1.00	0.33	1.00	1.00	1.00	0.33	1.00	1.00
7	1.00	0.17	0.00	0.00	1.00	0.17	1.00	0.33
8	NA	E	NA	E	1.00	0.33	1.00	0.33
9	NA	E	NA	E	1.00	0.33	1.00	0.33
10	NA	E	NA	E	1.00	0.33	1.00	0.33
	1.00	0.42	0.80	0.80	1.00	0.30	1.00	0.60

TABLE V
PRECISION AND RECALL AVERAGES FOR THE BEST EXTRACTORS

Alchemy		ParsCit		Union Extractors				Union Extractors	
Language		Title		Affiliations		Authors		Mails	
P	R	P	R	P	R	P	R	P	R
1	1	0.82	0.82	0.90	0.86	0.80	0.95	0.88	0.71

Case Study 2: We analyze the website of a Degree in Computer Science subject of our University. On this site, the system was able to retrieve all PDF documents that were in it. Below we present a comparison with the results obtained with respect to the other Case Study using the same extractors.

The ParsCit extractor performance is significantly better (around 30% in P and R) in Case 1 than in Case 2, in Title and Authors. This may be because of two reasons: on the one hand, in Case 2 there is a wider variety of document types (class presentations, notes, articles, etc.), which are less structured documents. On the other hand, most of the documents are in Spanish, where the effectiveness of the extractors used may be lower. The Recall of EI2 is greater in Case 2 than in Case 1, in a 24% for Authors and 22% for Affiliations. This indicates that on the website of Case 2 an increased number of entities are retrieved from related links.

In both cases, with the union of extractors, for Authors, Affiliations and emails, a significant improvement of Recall is achieved compared to the extractors that only consider the document text (ParsCit, PDFBox, regular expressions) without losing Precision. This is achieved by processing the neighboring webpages within the site where the object was found.

Table V shows the results of P and R averages over all the documents from both case studies using the best extractor for each field. It can be seen that in all cases they have been very successful in both Precision and Recall.

Some of the documents gathered in both case studies can be considered negative because they cannot be loaded into the repository. This is the case of documents used as class material but written by authors external to the institution (as in Case 2), or the resume of professors stored in the analyzed site (as in Case 1 and 2). Up to now the filtering of these documents is done manually, but the proposed system assist the repository administrator in the selection of plausible documents to be uploaded into the repository, providing him/her with a database containing the candidates EDOs and their affiliations data.

V. CONCLUSIONS

The proposed architecture automates the task of gathering documents within a restricted web domain in order to identify plausible educational digital objects to be uploaded into an institutional repository. We have implemented a prototype that from seed URLs generates a database with the metadata of interest. Then, the repository administrator decides which documents are EDOs written by professors of the institution, in order to invite them to deposit them in the repository. Even though negative cases may be collected (e.g. a resume or an article written by an external author), these cases can be easily discarded by the repository administrator as he/she has metadata information.

We have implemented a prototype that allows us to evaluate the feasibility of the proposed architecture and experiment the combination of extraction tools to achieve the best results in the fields of our interest.

With the union of the proposed extractors, we extract with good precision: Language, Titles, Authors, Affiliations and at least a contact email for each document.

The main contribution 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 family nodes. On the other hand, we deal with the gathering of diverse 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. Future work is intended to advance the automation of filtering negative cases, either by document category (e.g., CV) or by the affiliation of the authors if none of them belongs to the institution.

REFERENCES

[1] S. Martín, P. Bongiovani, P. Casali, and A. C. Deco, "Study on perspectives regarding deposit on open access repositories in the context of public universities in the central-eastern region of Argentina," *Scholarly Res. Commun.*, vol. 6, no. 1, pp. 1–13, Feb. 2015. [Online]. Available: http://src-online.ca/index.php/src/article/viewFile/145/389

[2] S. Albitar, B. Espinasse, and S. Fournier, "Combining agents and wrapper induction for information gathering on restricted Web domains," in *Proc. 4th Int. Conf. Res. Challenges Inf. Sci.*, Nice, France, May 2010, pp. 343–352.

[3] M. T. Pazienza, A. Stellato, and M. Vindigni, "Combining ontological knowledge and wrapper induction techniques into an e-retail system," in *Proc. Int. Workshop Tutorial Adapt. Text Extraction Mining (ATEM)*, Cavtat, Croatia, 2003, pp. 50–57.

[4] H. Li *et al.*, "CiteSeer: A scalable autonomous scientific digital library," in *Proc. 1st Int. Conf. Scalable Inf. Syst.*, Hong Kong, 2006, p. 18-es, doi: 10.1145/1146847.1146865.

[5] C. Castillo, "Effective Web crawling," Ph.D. dissertation, Dept. Comput. Sci., Univ. Chile, Santiago, Chile, Nov. 2004.

[6] L. Eikvil, "Information extraction from world wide Web—A survey," Norwegian Comput. Center, Oslo, Norway, Tech. Rep. 945, 1999.
[7] T. Pire, B. Espinase, A. Casali, and C. Deco, "Extracción automática de metadatos de Objetos de Aprendizaje: Un estudio comparativo," in *Proc. 4th Congreso Tecnología Educación Educación Tecnología*, Salta, Argentina, Jun. 2011, pp. 1–10.
[8] A. Casali, C. Deco, A. Romano, and G. Tomé, "An assistant for loading learning object metadata: An ontology based approach," *Interdiscipl. J. E-Learn. Learn. Objects*, vol. 9, pp. 77–87, Jan. 2013.
[9] A. Casali, C. Deco, C. Bender, F. Fontanarrosa, and C. Sabater, "Asistente para el depósito de objetos en repositorios con extracción automática de metadatos," in *Proc. Simposio Int. Tecnol. Inf. Comun. Edu.. (SINTICE)*, Madrid, Spain, 2013, pp. 133–136.
[10] P. Christen, "A comparison of personal name matching: Techniques and practical issues," Dept. Comput. Sci., Australian Nat. Univ., Canberra, ACT, Australia, Tech. Rep. TR-CS-06-02, 2006.
[11] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, no. 1, pp. 31–88, 2001.
[12] A. Monge and C. Elkan, "The field-matching problem: algorithm and applications," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 267–270.
[13] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Jun. 2013.

Ana Casali received the degree in mathematics from the Universidad Nacional de Rosario (UNR), Argentina, and the master's and Ph.D. degrees in information technologies from the University of Girona, Spain. She has been a Professor with the Facultad de Ciencias Exactas, Ingeniería y Agrimensura, UNR, since 1991, and the Head of the Computer Science Department since 2007. She is currently a Researcher with the Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas. She has worked in several international cooperation research projects. Her research interest includes agent architectures, knowledge representation, approximate reasoning, and recommender systems and their applications to education.

Claudia Deco received the degree in mathematics from the Universidad Nacional de Rosario (UNR), the master's degree in computer science from the Universidad de la República, Montevideo, Uruguay, and the Ph.D. degree in engineering from UNR. She is currently a Professor and Researcher with the Research Department, Facultad de Química e Ingeniería del Rosario, Universidad Católica Argentina Sede Rosario. Furthermore, she coordinates the Research Group of Databases with the Facultad de Ingeniería, UNR. Her research interest includes databases technologies and information retrieval.

Santiago Beltramone received the Computer Science degree from the Universidad Nacional de Rosario (UNR), Argentina. He is currently a member of research projects with the Facultad de Ciencias Exactas, Ingeniería y Agrimensura, UNR. He is also a Software Developer at a local enterprise.