Learning to Harvest Information for the Semantic Web

Fabio Ciravegna, Sam Chapman, Alexiei Dingli, and Yorick Wilks

Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street, S1 4DP Sheffield, UK
N.Surname@dcs.shef.ac.uk
http://nlp.shef.ac.uk/wig/

Abstract. In this paper we describe a methodology for harvesting information from large distributed repositories (e.g. large Web sites) with minimum user intervention. The methodology is based on a combination of information extraction, information integration and machine learning techniques. Learning is seeded by extracting information from structured sources (e.g. databases and digital libraries) or a user-defined lexicon. Retrieved information is then used to partially annotate documents. Annotated documents are used to bootstrap learning for simple Information Extraction (IE) methodologies, which in turn will produce more annotation to annotate more documents that will be used to train more complex IE engines and so on. In this paper we describe the methodology and its implementation in the Armadillo system, compare it with the current state of the art, and describe the details of an implemented application. Finally we draw some conclusions and highlight some challenges and future work.

1 Introduction

The Semantic Web (SW) needs semantically-based document annotation to both enable better document retrieval and empower semantically-aware agents. Most of the current technology is based on human centered annotation, very often completely manual. The large majority of SW annotation tools address the problem of single document annotation. Systems like COHSE [7], Ontomat [8] and MnM [14], all require presenting a document to a user in order to produce annotation either in a manual or a (semi-)automatic way. Annotations can span from annotating portions of documents with concept labels, to identifying instances or concept mentions, to connect sparse information (e.g. a telephone number and its owner. The process involves an important and knowledge intensive role for the human user. Annotation is meant mainly to be statically associated to the documents. Static annotation can: (1) be incomplete or incorrect when the creator is not skilled enough; (2) become obsolete, i.e. not be aligned with page updates; (3) be devious, e.g. for spamming or dishonest purposes; professional spammers could use manual annotation very effectively for their own purposes.

For these reasons, we believe that the Semantic Web needs automatic methods for (nearly) completely automatic page annotation. In this way, the initial annotation associated to a document will lose its importance because at any time it will be possible to automatically reannotate the document. Systems like SemTag [4] are a first step in that direction. SemTag addresses the problem of annotating large document repositories (e.g. the Web) for retrieval purposes, using very large ontologies. Its task is annotating portion of documents with instance labels. The system can be seen as an extension of a search engine. The process is entirely automatic and the methodology is largely ontology/application independent. The kind of annotation produced is quite shallow when compared to the classic one introduced for the SW: for example there is no attempt to discover relations among entities. AeroDaml [9] is an information extraction system aimed at generating draft annotation to be refined by a user in a similar way to nowadays' automated translation services. The kind of annotation produced is more sophisticated than SemTag's (e.g. it is also able to recognize relations among concepts), but, in order to cover new domains, it requires the development of application/domain specific linguistic knowledge bases (an IE expert is required). The harvester of the AKT triple store[1] is able to build large knowledge bases of facts for a specific application. Here the aim is both large scale and deep ontology-based annotation. The process requires writing a large number of wrappers for information sources using Dome, a visual language which focuses on manipulation of tree-structured data [11]. Porting requires a great deal of manual programming. Extraction is limited to highly regular and structured pages selected by the designer. Maintenance is complex because - as well known in the wrapper community - when pages changes their format, it is necessary to re-program the wrapper [10]. The approach is not applicable to irregular pages or free text documents. The manual approach makes using very large ontologies (like in SemTag) very difficult.

In this paper we propose a methodology for document annotation that was inspired by the latter methodology, but (1) it does not require human intervention for programming wrappers (2) it is not limited to highly regular documents and (3) it is largely unsupervised. The methodology is based on adaptive information extraction and integration, it is implemented in Armadillo, a tool able to harvest domain information from large repositories. In the rest of the paper we describe and discuss the methodology, present experimental results on a specific domain and compare Armadillo with the current state of the art. Finally we outline some challenges that the methodology highlights.

2 Armadillo

Armadillo is a system for producing automatic domain-specific annotation on large repositories in a largely unsupervised way. It annotates by extracting information from different sources and integrating the retrieved knowledge into a repository. The repository can be used both to access the extracted information and to annotate the pages where the information was identified. Also the link

[1] http://triplestore.aktors.org/SemanticWebChallenge/

```
Input:
    •an Ontology;
    •an Initial Lexicon;
    •a Repository of Documents;
Output: A set of triples representing the extracted information
        and to be used to annotate documents
do {
    ▪ spot information using the lexicon
    ▪ seek for confirmation of the identified information
    ▪ extend lexicon using adaptive information extraction
      and seek confirmation of the newly extracted information
    } while a stable set of information is found(e.g. the base
              does not grow anymore)
• Integrate Information from different documents
• Store information in repository
```

Fig. 1. The Armadillo's main algorithm

with the pages can be used by a user to verify the correctness and the provenance of the information. Armadillo's approach is illustrated in Figure 1. In the first step in the loop, possible annotations from a document are identified using an existing lexicon (e.g. the one associated to the ontology). These are just potential annotations and must be confirmed using some strategies (e.g. disambiguation or multiple evidence). Then other annotations not provided by the lexicon are identified e.g. by learning from the context in which the known ones were identified. All new annotations must be confirmed and can be used to learn some new ones as well. They will then become part of the lexicon. Finally all annotations are integrated (e.g. some entities are merged) and stored into a data base. Armadillo employs the following methodologies:

– Adaptive Information Extraction from texts (IE): used for spotting information and to further learning new instances.
– Information Integration (II): used to (1) discover an initial set of information to be used to seed learning for IE and (2) to confirm the newly acquired (extracted) information, e.g. using multiple evidence from different sources. For example, a new piece of information is confirmed if it is found in different (linguistic or semantic) contexts.
– Web Services: the architecture is based on the concept of "services". Each service is associated to some part of the ontology (e.g. a set of concepts and/or relations) and works in an independent way. Each service can use other services (including external ones) for performing some sub-tasks. For example a service for recognizing researchers names in a University Web Site will use a Named Entity Recognition system as a sub-service that will recognise potential names (i.e. generic people's names) to be confirmed using some

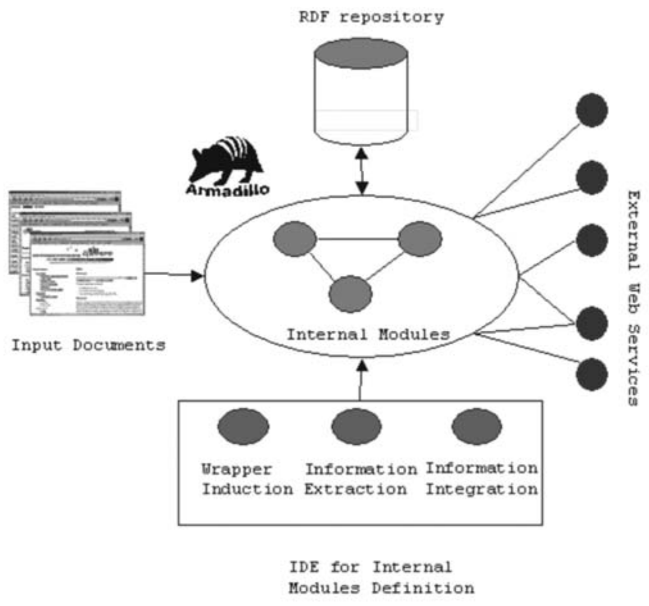

Fig. 2. The Armadillo Architecture

internal strategies as real researchers names (e.g. as opposed to secretaries' names).
- RDF repository: where the extracted information is stored and the link with the pages is maintained.

A development environment allows to define architectures for new applications. Porting to new applications does not require knowledge of IE. All the methods used tend to be domain independent and are based on generic strategies to be composed for the specific case at hand. The only domain dependent parts are: the initial lexicon, the ontology and the way the confirmation strategies are designed/composed.

2.1 Extracting Information

Most of the current tools (e.g. COHSE and SemTag) provide annotation using a static lexicon where the lexicalization of objects in the ontology is contained. The lexicon does not increase while the computation goes on, unless the user adds terms to it. Armadillo continually and automatically expands the initial lexicon by learning to recognize regularities in the repository. As a matter of fact, Web resources (and in general all repositories) have a specific bias, i.e. there are a number of regularities, either internal to a single document or across a set of documents [13]. Regularities can either be very strong (e.g. in case of pages generated by a data base), or given by a style imposed by the designers. Armadillo is able to capture such regularities and use it to learn to expand its initial lexicon. There are two ways in which an object (e.g., an instance of a concept) can be identified in a set of documents. Using its internal description (e.g. its name or the words describing it) and the context in which it appears. Systems like COHSE, SemTag and Magpie [6] use the former. MnM and Ontomat [8] use adaptive IE to learn from the context as well. They use the regularity in a collection of documents (e.g. a set of web pages about a specific topic from the same site) to derive corpus-wide rules. In those approaches, it is very important that the corpus is carefully chosen as consistent in its regularity. This allows learning from the human annotation to converge quickly to a stable and effective situation. Armadillo uses an approach that can be seen as an extension of the one used in MnM and Ontomat where there is no human in the loop and where large diverse repositories (e.g. whole portions of the Web) are annotated, and therefore such regularity is not straightforward. The system has to find its own ways to identify those regularities and use them to learn without user support. When regularities in the context are found, they are used to learn other occurrences of the same (type of) object.

2.2 Gradually Acquiring Information

Armadillo exploits a key feature of the Web: the *redundancy* of information. Redundancy is given by the presence of multiple citations of the same information in different contexts and in different superficial formats. Redundancy is currently used for improving question answering systems [5]. When known information is present in different sources, it is possible to use its multiple occurrences to bootstrap recognizers that, when generalized, will retrieve other pieces of information, producing in turn more (generic) recognizers [1]. Armadillo uses redundancy in order to bootstrap learning beyond the initial user lexicon (if any) and even to acquire the initial lexicon. In particular, information can be present in different formats on the Web: in documents, in repositories (e.g. databases or digital libraries), via agents able to integrate different information sources, etc. From them or their output, it is possible to extract information with different degrees of reliability. Systems such as databases contain structured data that can be queried either via APIs or web front ends (getting HTML output). In the latter case, wrappers can be induced to extract information. Wrapper Induction methodologies are able to model rigidly structured Web pages such as those produced by databases [10]. When the information is contained in textual documents, extracting information requires more sophisticated methodologies. Wrapper induction systems have been extended to cope with less rigidly structured pages, free texts and even a mixture of them [2]. There is an increasing degree of complexity in the extraction task mentioned above. As complexity increases more training data is required. Wrappers can be trained with a handful of examples whereas full IE systems may require millions of words.

All the IE process in Armadillo is based on integrating information from different sources to provide annotations which will bootstrap learning, which in turn will provide more annotation and so on. The process starts with simple methodologies which require limited annotation, to produce further annotation to train more complex modules. The ontology provides the mean for integrating information extracted from different sources. For example simple wrappers can be used to extract information from a web page produced by databases containing papers from computer science departments. In order to avoid wrapping each database separately (i.e., providing examples of annotated input/output for each of them), Armadillo uses information from a database already wrapped in order to provide automatic annotation of examples for the other ones as proposed in [13]. For example, if the goal is to extract bibliographic information about the Computer Science field, it is possible to use Citeseer (www.citeseer.com), a large (and largely incomplete) database to learn how to query and understand another service, e.g. the NLDB bibliography at Unitrier (http://www.informatik.uni-trier.de/∼ ley/db/). This can be done by querying Citeseer and the NLDB using the same terms, and producing two parallel pages of results. The one from Citeseer will have a known format and the information can be easily extracted using a predefined wrapper. Then, some of the information contained in the NLDB output page can be automatically annotated (e.g. for the paper title generally it is necessary just an intelligent string matching). Using the annotated examples it is possible to induce wrappers that, given the high regularity of the information in the NLDB page, will be able to extract papers also from the latter. Considering that training a wrapper generally requires just a handful of examples, it is possible to focus only on those examples where the match is very clear and reliable, discarding others that are more questionable, therefore producing a highly reliable wrapper. Facilities for defining wrappers are provided in our architecture by Amilcare (nlp.shef.ac.uk/amilcare/), an adaptive IE system based on a wrapper induction methodology able to cope with a whole range of documents from rigidly structured documents to free texts [3].

2.3 Web Services

Each task in Armadillo (e.g. discovering all the papers written by an author) is performed by a server which in turn will use other servers for implementing some parts (subtask) of it. Each server exposes a declaration of input and output, plus a set of working parameters. Servers are reusable in different contexts and applications. For example one server could return all papers written by a person by accessing Citeseer. Another one will do the same on another digital library. Another one will use the papers extracted by the other two in order to discover pages where they are cited and to bootstrap learning. Another one will invoke these servers and integrate the evidence returned by each of them and decide if there is evidence enough to conclude that some newly discovered candidate strings represent a real new object or maybe just a variant version of a known name. All the servers are defined in a resource pool and can be used in a user-defined architecture to perform some specific tasks. New servers can be defined and added to the pool by wrapping them in a standard format. The defined architecture works as a "Glass Box". All the steps performed by the system are shown to the user together with their input and output. The user can check the intermediate results and manually modify their output, or change their strategy (if possible, such as in the case of modules who integrate information). For example if a piece of information is missed by the system, it can be manually added by the user. The modules working on the output of that module will then be re-run and further information will hopefully be retrieved. In this way the user is able both to check the results of each step and to improve the results of the system by manually providing some contributions (additions, corrections, deletion).

2.4 The Triple Store

Facts extracted from the Web are stored in an RDF store in the form of triples which define relations in the form "Subject - Verb - Object", where the subject and object are elemenets and the verb details the relation between them. For each element in the triples, the following information is stored: the string (e.g. J. Smith), the position where it was found (e.g. the document URL and its offset) and the concept, instance or relation represented. The triples can be used to derive also aliases for the same object, i.e. a lexicon ("J. Smith" at $< www\_address_1 >: 33 : 44$ and "John Smith" at $< www\_address_2 >: 21 : 35$), and to recover dispersed information (e.g. the person JSMITH45 has names "J. Smith" at $< www\_address_1 >: 33 : 44$ and "John Smith" at $< www\_address_2 >: 21 : 35$ and telephone number "+44.12.12.12.12 at $< www\_address_3 >: 10 : 12$, homepage at $< www\_address_4 >$). The triple store constitutes the resource used by the different services to communicate. Each server stores the extracted information in the form of signed triples. The other services will extract them and elaborate the information to store further information (or to confirm the existing one). Each piece of information is tagged with its provenance both in terms of source document and in terms of extraction method, i.e. the service or agent that has retrieved it. The provenance is used to assign reliability to the information itself: the more the information is confirmed, the more reliable it is considered.

2.5 Confirming Information

A crucial issue in the cycle of seeding and learning is the quality of the information used to seed. Wrong selections can make the method diverge and produce spurious information. When a piece of information is acquired (e.g. a new paper is assigned to a specific author in the CS task mentioned in section 3), Armadillo requires confirmation by different sources before it can be used for futher seeding of learning. Again, using the redundancy of the Web, we expect that the same piece of information is repeated somewhere in a different forms and the system to find it. The strategy for evidence gathering is application dependent. Users have to identify task specific strategies. Such strategies can be defined declaratively in Armadillo by posing requirements on the provenance of the information in terms of methods of extraction and sources (including the number of times the information was confirmed in the different sources).

The next section describes how Armadillo was used in a specific application.

3 Armadillo in Action

Armadillo has been applied so far to three tasks: the CS website harvesting task, the Art Domain task and the discovery of geographical information[2]. Here we describe Armadillo's application to mining websites of Computer Science Departments, an extension of its original task in the AKTive Space application that won the 2003 Semantic Web Challenge[3]. Armadillo's task is to discover who works for a specific department (name, position, homepage, email address, telephone number) and to extract for each person some personal data and a list of published papers larger than the one provided by services such as Citeseer.

3.1 Finding People Names

The goal of this subtask is to discover the names of all the people who work in the specific department. This task is more complex than a generic Named Entity Recognition because many non researchers' names are cited in a site, e.g. those of undergraduate students, clerics, secretaries, etc, as well as names of researchers from other sites that e.g. participate in common projects or have co-authored papers with members of staff. Organizing the extraction around a generic Named Entity Recognizer (NER) is the most natural option. This does not finish the job, though, because a NER recognizes ALL the people's names in the site, without discriminating between relevant and irrelevant. Moreover classic NERs tend to be quite slow if launched on large sites and can be quite imprecise on Web pages, as they are generally defined for newspaper-like articles. A two-step strategy is used here instead: initially a short list of highly reliable seed names are discovered; this constitute the initial lexicon. Such lexicon could also be provided by an existing list. Here we suppose such list does not exist. Then these seeds are used to bootstrap learning for finding further names.

Finding Seed Names. To find seed names, a number of weak strategies are combined that integrate information from different sources. First of all the web site is crawled looking for strings that are potential names of people (e.g. using a gazetteer of first names and a regular expression such as <first-name<$^+$(capitalized word)$^+$.). Then the following web services are queried:

– Citeseer (www.citeseer.com): Input: the potential name; Output: a list of papers and a URL for homepage (if any);

[2] http://www.dcs.shef.ac.uk/~sam/results/index.html
[3] http://challenge.semanticweb.org/)

- The CS bibliography at Unitrier (http://www.informatik.uni-trier.de/∼ ley/db/): Input: the potential name: Output: a list of papers (if any);
- HomePageSearch (http://hpsearch.uni-trier.de/): Input: the potential name; Output: a URL for homepage (if any);
- Annie (www.gate.ac.uk): Input: the potential name and the text surrounding it; Output: True/False;
- Google (www.google.co.uk) Input: the potential name and the URL of the site in order to restrict search; Output: Relevant Pages that are hopefully homepages;

The digital libraries (Citeseer and Unitrier) are used as first filters to determine if a string is a name of a known researcher. If they return reasonable results for a specific string (i.e. not too few and not too many), this name is further processed, otherwise it is discarded. A string is a potentially valid name if the digital libraries return a reasonable number of papers (between 5 and 50 in our experiments). Results not in line with the reasonability criteria are discarded as inappropriate seeds. This is to discard potential anomalies such as ambiguous names (e.g. Citeseer returns more than 10,000 papers for the term "Smith"; this cannot be a single researcher) and invalid names (e.g. the words "Fortune Teller" do not return any paper). We tend to use quite restrictive criteria for keeping reliability high (i.e. it is very possible that a person writes some 100 papers, but that amount could also hide name ambiguity. The results of the digital libraries are integrated with those of the classic Named Entity Recognizer run on a window of words around the candidates (so to avoid the problem of slow processing). At this point a number of names are available that fall in three potential types: (1) correct (they are people working for the department); (2) wrong (they are not people: they are false positives); (3) people who do not work at the site but that are cited because, for example, they have coauthored papers with some of the researchers of the department. For this reason, Citeseer, Google and HomepageSearch are used to look for a personal web page in the site. If such a page is not found, the names are discarded. From the results, personal web pages are recognized with simple heuristics such as looking for the name in the title or in "<H1>" tags. The process mentioned above is meant to determine a small, highly reliable list of seed names to enable learning. Each of the strategies is, per se, weak, as they all report high recall, low precision. Their combination is good enough to produce data with high accuracy.

Learning Further Names. All the occurrences of seed names are then annotated on the site's documents and learning is initiated only on documents where a reasonable quantity of known names are organized in structures such as lists and tables. Such structures generally have an intrinsic semantic: lists generally contain elements of the same type (e.g. names of people), while the semantics in tables is generally related to the position either in rows or columns (e.g. all the elements of the first column are people, the second column represents addresses, etc.). When some seeds (at least four or five in our case) are identified in a list or specific portions of a table, we train a classifier able to relate a large part of these examples, for example using linguistic and/or formatting criteria (e.g. relevant names are always the first element in each row). If we succeed, we are able to reliably recognize other names in the structure. Every department generally has one or more pages listing their staff in some kind of lists. These are the lists that we are mainly looking for, but also tables assigning supervisors and students are useful, provided that students and teachers can be discriminated. Each time new examples are identified, the site is further annotated and more patterns can potentially be learnt. New names can be cross-checked on the resources used to identify the seed list: we have now more evidence that these names are real names. In our experiments this is enough to discover a large part of the staff of an average CS website with very limited noise, even using a very strict strategy of multiple cross-evidence. We are currently using combinations of the following evidence to accept a learnt name: (1) the name was recognized as seed; (2) the name is included in an HTML structure where other known occurrences are found (3) there is an hyperlink internal to the site that wraps the whole name; (4) there is evidence from generic patterns (as derived by recognizing people on other sites) that this is a person. The latter strategy was inspired by [12].

Evaluation. The CS task effectiveness was evaluated on a number of sites. Here we report results from an evaluation done on both a specific web site (the Computer Science Department site of the University of Sheffield, www.dcs.shef.ac.uk) and by pointing the system to some pages containing interesting infromation for the task, but distributed in random sites (the latter task is equivalent to applying Armadillo on the results of a document classifier providing interesting pages). Results on other sites are qualitatively largely equivalent. On the Sheffield department's website, the system initially discovers 51 seed names of people (either academics, researchers or PhD students) integrating information from Citeseer and NLDB. Of them, 48 are correct and 3 wrong. These names are used to the seed learning. Learning allows to discover other 57 names, 48 correct, 6 wrong. This increases the overall recall from 37% to 84% with a very limited loss in precision (see Table 1). Results obtained on the set of pages from random sites are in Table 2. In this experiments we checked Armadillo's ability to improve results in case the II step returned high recall. The AKT triple store was used as a source of information in addition to Citeseer and UniTrier. The gain in using the IE-based extraction is still considerable (recall grows from 73 to 85).

3.2 Discovering Papers Citations

Discovering what papers are written by what members of the departmental staff is a very difficult task. It requires recognizing the paper title and the authors, and then relating the authors to the people identified in the previous step. Authors are names in particular positions and in particular contexts: they must not be confused with editors of collections in which the paper can be published, nor they must be confused with other names mentioned in the surrounding text. A title is generally a random sequence of words (e.g. the tile of [5]) and cannot be

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
Seed discovery	129	51	48	3	0	94	37	51
Adaptive IE	129	108	99	9	30	92	84	87

Table 1. Results in Discovering People and Associated homepage. First line: accuracy reached using Information Integration only (Citeseer+Google, etc.); second line: accuracy using adaptive IE. Possible represents the number of people working for the department, Actual the number of people returned by the systems. Actual results are divided into Correct, Wrong and Missing.

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
Seed Discovery	331	243	242	1	89	99.59	73.11	84.32
Adaptive IE	331	288	284	4	47	98.61	85.80	91.76

Table 2. The results of discovering names of peoples working at some random sites.

characterized in any way (i.e. we cannot write generic patterns for identifying candidate strings as we did for people). Moreover paper titles must not be confused with titles of collections in which they are published. CS department sites typically contain lists of publications for the department as a whole or personal ones for each member of staff. Moreover papers are co-authored, so it is very possible that each paper is cited more than one time within a specific site. In rare cases personal lists of papers are produced using a departmental database (i.e. all the publication pages are formatted in the same way), but in most cases each person writes the list using a personal format; very often the style is quite irregular as the list is compiled manually over time. This is a typical case in which the classic methodology of manually annotating some examples for each page for each member of staff is unfeasible, due to the large number of different pages. Also irregularities in style produce noisy data and classic wrappers are not able to cope with noise. A generic methodology is needed that does not require any manual annotation.

In order to bootstrap learning we query the digital libraries (Citeseer and UniTrier) using staff names as keywords. The output for each name is hopefully a list of papers. Such lists will be incomplete because the digital libraries are largely incomplete. The titles in the list are then used to query a search engine to retrieve pages containing multiple paper citations. We focus on lists and tables where at least four papers are found. We use titles because they tend to be unique identifier. We are looking for seed examples, so we can discard titles which report too many hits (so to avoid titles which are very common strings such as "Lost"). As for discovering new papers, the seed examples are annotated and page-specific patterns are induced. We favour examples contained in structures such as lists

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
Seed Discovery	320	151	152	1	168	99	47	64
Adaptive IE	320	217	214	3	103	99	67	80

Table 3. Paper title harvesting for 7 random people.

and tables for which we have multiple evidence. Please note however that the structure of the citation is often not very structured internally. For example:

```
<li> Fabio Ciravegna, Alexiei Dingli, Daniela Petrelli and Yorick Wilks:<br>
User-System Cooperation in Document Annotation based on Information Extraction
<br> in Asuncion Gomez-Perez, V. Richard Benjamins (eds.): Knowledge
Engineering and Knowledge Management (Ontologies and the Semantic Web),
<br> Lecture Notes in Artificial Intelligence 2473, Springer Verlag <br></li>
```

Simple wrappers would be ineffective, as there is no way to discriminate - for example - between authors and editors and title of paper and title of collection when relying on the HTML structure only. More sophisticated wrapper induction systems are needed, as that provided by Amilcare, which uses both html structure and (para-)linguistic information [3]. Using a cycle of annotation/learning/annotation we are able to discover a large number of new papers. Note that every time co-authorship among people is discovered on a publication page of one author, the paper is retained for annotation of the publication pages of the other authors (i.e. the redundancy is exploited again).

Evaluation. Discovering papers is a very complex task. We performed a task of associating papers to people discovered during the previous step. A paper was considered correctly assigned to a person if it was authored by the person **and** the title was 100% correct. We did not use reseeding in the experiment, i.e. if a paper was coauthored by two researchers, the information returned for one person was not used to further annotate the publication pages for the second person. In this sense the redundancy of information was not fully exploited. Checking correctness of papers for a hundred people is very labor intensive, therefore we randomly checked the papers extracted for 7 staff members for which the seed papers exceeded 6 examples; results are shown in Table 3. The use of IE increases significantly the overall recall rate which grows from 47 for seeds and 67 for IE-based, precision 99 and 98 and F-measure 64 and 80 respectively.

4 Conclusions

In this paper we have described a methodology to extract information from large repositories (e.g. large Web sites) with minimum user intervention. Extracted information can then be used for document annotation. Information is initially extracted by starting from highly reliable/easy-to-mine sources such as databases and digital libraries and is then used to bootstrap more complex modules such as wrappers for extracting information from highly regular Web pages. Information extracted by the wrappers is then used to train more sophisticated IE engines. All the training corpora for the IE engines are produced automatically. Experiments show that the methodology can produce high quality results. The user intervention is limited to providing an initial URL and to add information missed by the different modules when the computation is finished. No preliminary manual annotation is required. The information added or deleted by the user can then be reused for restarting learning and therefore getting more information (recall) and/or more precision. The type of user needed is a person able to understand the annotation task. No skills in IE are needed. The natural application of such methodology is the Web, but large companies' repositories are also an option. In this paper we have focused on the use of the technology for mining web sites, an issue that can become very relevant for the Semantic Web, especially because annotation is provided largely without user intervention. It could potentially provide a partial solution to the outstanding problem of who is providing semantic annotation for the SW. It can potentially be used either by search engines associated to services/ontologies to automatically annotate/index/retrieve relevant documents or by specific users to retrieve needed information on the fly by composing an architecture.

Armadillo has been fully integrated into the AKT triple store and it is constantly providing new triples to it. Its contribution to the architecture is the ability to reindex the pages when they change format (in the classic architecture this step would require manually reprogramming of the wrapper) and the abillity to extract information from sources that are not highly structured and regular as possible to Dome [11]. Armadillo is compatible with SW tools like COHSE, Magpie, MnM, Ontomat, etc. In COHSE and Magpie it could provide a way to (1) extend the automatic annotation step beyond the connection between simple terms and concepts descriptions stored in a lexicon. It could allow to move towards relation identification. Moreover it couls provide automatic extension of the initial lexicon. In MnM and Ontomat, Armadillo could provide a way to converge more rapidly towards an effective annotation service. As a matter of fact, learning in those tools is limited to the documents already annotated by the user and to the use of an initial lexicon. Armadillo could provide a way to integrate information from external repositories in the corpus (e.g. digital libraries) to learn in an unsupervised way, from example from regularities found in documents not annotated.

From the IE point of view there are a number of challenges in learning from automatic annotation, instead of using human annotation. On the one hand not all the annotation is reliable: the use of multiple strategies and combined evidence reduces the problem, but still there is a strong need for methodologies robust with respect to noise. On the other hand, many IE systems are able to learn from completely annotated documents only, so that all the annotated strings are considered positive examples and the rest of the text is used as a set of counterexamples. In our cycle of seed and learn, we generally produce partially annotated documents. This means that the system is presented with positive examples, but the rest of the texts can never be considered as a set of negative examples, because unannotated portions of text can contain instances that the system has to discover, not counterexamples. This is a challenge for the learner. At the moment we present the learner with just the annotated portion of the text plus a windows of words of context, not with the whole document. This is enough to have the system learning correctly: the unannotated examples that become negative examples entering the training corpus is generally low enough to avoid problems. In the future we will have to focus on using machine learning methodologies that are able to learn from scattered annotation.

Many of the classic problems of integrating information are to be coped with in Armadillo. Information can be represented in different ways, in different sources from both a syntactic and a semantic point of view. The syntactic variation is coped with in the definition architecture definition step: when two modules are connected, a canonical form of the information is defined in the ontology, e.g. the classic problem of recognising film titles as "The big chill" and "Big chill, the" can be addressed. More complex tasks are to be addressed, though. In the art domain it is quite common to report the title of an art work in different languages. For example a number of Cezanne's paintings can be referred in different web sites as both "Apples and Oranges" and "Aepfel mit Orangen" (same title but in German), Michelangelo's can be referred as "The Last Judgment" or "Il Giudizio Universale" (in Italian). Relating them can be very difficult, even for a human, without looking at the actual artwork. Also, a person name can be cited in different ways in different documents: N. Weaver, Nick Weaver and Nicholas Weaver are potential variation of the same name. But do they identify the same person as well? This is the problem of intra- and inter-document coreference resolution well known in Natural Language Processing. In many applications it is possible to identify some simple heuristics to cope with this problem. For example in mining one specific CS websites, N. Weaver, Nick Weaver and Nicholas Weaver are most of the times the same person, therefore it is possible to hypothesize coreference. Another potential problem concerns ambiguity in the external resources (e.g. in the digital libraries). When querying with very common names (e.g. "John Smith") papers by different people are mixed. This is not a problem in Armadillo because the information returned is used to annotate the site in order to both seed more learning and to look for multiple confirmation. Papers from people from other departments or universities will not introduce any annotations and therefore will not be accepted. The same applies in case multiple homepages are returned: if some of them do not have an address local to the current site, they are not used.

Acknowledgements

This work was carried out within the AKT project (www.aktors.org), sponsored by the UK Engineering and Physical Sciences Research Council (grant GR/N15764/01), and the Dot.Kom project (www.dot-kom.org), sponsored by the EU IST asp part of Framework V (grant IST-2001-34038).

References

1. Sergey Brin. Extracting patterns and relations from the world wide web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98*, 1998.
2. Fabio Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2001. Seattle.
3. Fabio Ciravegna. Designing adaptive information extraction for the Semantic Web in Amilcare. In S. Handschuh and S. Staab, editors, *Annotation for the Semantic Web*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2003.
4. S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the World Wide Web Conference 2003*, 2003.
5. Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. Web question answering: Is more always better? In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, Tampere, Finland, 2002.
6. Martin Dzbor, John B. Domingue, and Enrico Motta. Magpie - towards a semantic web browser. In *Proceedings of the 2nd Intl. Semantic Web Conference*, October 2003. Sanibel Island, Forida.
7. C. Goble, S. Bechhofer, L. Carr, D. De Roure, and W. Hall. Conceptual Open Hypermedia = The Semantic Web? In *The Second International Workshop on the Semantic Web*, pages 44–50, Hong Kong, May 2001.
8. S. Handschuh, S. Staab, and F. Ciravegna. S-CREAM - Semi-automatic CREAtion of Metadata. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02*. Springer Verlag, 2002.
9. P. Kogut and W. Holmes. Applying information extraction to generate daml annotations from web pages. In *Proceedings of the K-CAP 2001 Workshop Knowledge Markup & Semantic Annotation*, 2001. Victoria B.C., Canada.
10. N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997.*, 1997.
11. Thomas Leonard and Hugh Glaser. Large scale acquisition and maintenance from the web without source access. In Siegfried Handschuh, Rose Dieng-Kuntz, and Steffen Staab, editors, *Proceedings Workshop 4, Knowledge Markup and Semantic Annotation, K-CAP 2001*, 2001.
12. Tom Mitchell. Extracting targeted data from the web. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, San Francisco, California, 2001.
13. M. Perkowitz and O. Etzioni. Category translation: Learning to understand information on the internet. In *International Joint Conference on Artificial Intelligence, IJCAI-95*, pages 930–938, Montreal, Canada, 1995.
14. M. Vargas-Vera, Enrico Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology driven semi-automatic or automatic support for semantic markup. In *Proc. of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02*. Springer Verlag, 2002.