Automatic Semantic Annotation using Unsupervised Information Extraction and Integration

Alexiei Dingli, Fabio Ciravegna, and Yorick Wilks

ABSTRACT

In this paper we propose a methodology to learn to automatically annotate domain-specific information from large repositories (e.g. 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). Retrieved information is then used to partially annotate documents. These annotated documents are used to bootstrap learning for simple Information Extraction (IE) methodologies, which in turn will produce more annotations used to annotate more documents. It will be used to train more complex IE engines and the cycle will keep on repeating itself until the required information is obtained. The user intervention is limited to providing an initial URL and to correct information if it is the case when the computation is finished. The revised annotation can then be reused to provide further training and therefore getting more information and/or more precision.

1. INTRODUCTION

The Semantic Web (SW) needs semantically-based document annotation[1] 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 [11]. Manual annotation is difficult, time consuming and expensive [3]. Convincing millions of users to annotate documents for the Semantic Web is difficult and requires a world-wide action of uncertain outcome. In this framework, annotation is meant mainly to be statically associated to (and saved within) the documents. Static annotation associated to a document can: (1) be incomplete or incorrect when the creator is not skilled enough; (2) become obsolete, i.e. not be aligned with pages updates; (3) be irrelevant for some use(r)s: a page in a pet shop web site can be annotated with shop-related annotations, but some users would rather prefer to find annotations related to animals.

Different annotation can be imposed on a document using different ontologies. An ontology is required because it describes concepts and relationships that occur in a very restricted view of the real world, basically it describes the domain in which we are working. Most of the annotation is likely to be associated in the future by Web actors other than the page's owner, exactly like nowadays' search engines produce indexes without modifying the page code. Producing methodologies for automatic annotation of pages with no or minimal user intervention becomes therefore important: the initial annotation associated to the document loses its importance because at any time it is possible to automatically (re)annotate the document and to store the annotation in a separate database or ontology. In the future Semantic Web, automatic annotation systems might become as important as indexing systems are nowadays for search engines.

Automatic annotation methodologies have been developed in the past at the intersection of research areas such as Information Extraction from text (IE)[19], Information Integration (II)[12], wrapper induction [13] and machine learning [17]. Porting to new application domains in fairly unconstrained areas such as the Web is out of reach for the current technologies. The association to domain-specific ontologies limits the domain and makes the application feasible. For example IE is currently used to reduce the burden in some SW annotation tools[20] [10] [3] and used to crawl the Web for harvesting domain specific information [14] [17]. Most of this technology is based on supervised learning, i.e. they require user-defined annotated corpora. Producing such corpora can be difficult and time consuming, even when using adaptive IE as support to annotation. If we take a look at II, we find that there are many projects focusing on the web[9] [15] [12] but they all fall short when it comes to process sites with very irregular formatting or containing free text. If the documents are many and very different, and the annotation to be performed very detailed, the annotation process may require a substantial work. In case of annotating large portions of the Web (e.g. groups of whole web sites), the task could become unfeasible. Therefore, there is the need of training the system without manually producing the annotated material.

In this paper we propose a methodology to learn how to annotate semantically-consistent portions of the Web extracting and integrating information from different sources. All the annotation is produced automatically with no user

[1]Semantic annotation is the process of inserting tags in the document, whose purpose is to assign semantics to the text between the opening and closing tags.

intervention apart some corrections the users might want to perform. The methodology has been fully implemented in Armadillo[2], a system for unsupervised information extraction and integration from large collections of documents. The natural application of such methodology is the Web, but large companies' information systems are also an option. In this paper we will focus on the Web, and in particular in mining web sites of Computer Science Departments. All the process is based on integrating information from different sources in order to provide some seed annotations. This will then bootstrap learning which in turn will provide more annotations and so on. In synthesis we start with a simple methodology which requires limited annotation, and move on to produce further annotation to train more complex modules.

In the next section we will present the generic architecture that we have used to build the application. Then we will describe the CS Department task and how the information from different sources is integrated to learn to annotate the desired information. Experimental results are illustrated and finally we will discuss some future challenges that our experience highlights.

2. ARMADILLO: A GENERIC ARCHITECTURE FOR WEBSITE MINING

Since there can be many potential applications of the proposed technology we decided to make the Architecture as generic as possible. In order to do so, the Architecture had to be portable and scalable. Portability was achieved by making use of a simple methodology. The system starts from a set of generic strategies defining where to search information and what to look for. When data is harvested using those strategies, it is passed to an oracle in order to verify whether it is valid information or not. By oracle we mean a human or a resource (such as an IE engine, a database etc.) that can identify which items on a web page are instances of concepts found in a preloaded ontology. Once we have some information, the system applies further strategies which can be used with that kind of information and the results are given to other more specialised oracles. Basically the system keeps on looping until there is no more information to discover or the user decides to interrupt the cycle. It is clear that we don't make any assumptions on any domain. The strategies range from simple/generic ones (like looking for capitalised words) to more complex ones (like setting up and using IE tools) and they can be easily created for any domain. This technique works because a key feature of the web is 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. This factor is currently used for improving question answering systems [7]. 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 [2]. 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 reliability. Systems such as databases generally contain structured data and can be queried using an API. In case the API is not available (e.g. the database has a web front end and the output is textual), wrappers can be induced to extract such information. Wrapper Induction methodologies are able to model rigidly structured Web pages such as those produced by databases [13] [18]. 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 [8], free texts and even a mixture of them [4]. There is an obvious increasing degree of complexity in the extraction task mentioned above. The more the task is difficult, the less reliable generally the extracted information is. For example wrapper induction systems generally reach 100% on rigidly structured documents, while IE systems reach some 70% on free texts. Also, the more the complexity increases, the more the amount of data needed for training grows: wrappers can be trained with a handful of examples whereas full IE systems may require millions of words [16]. This is just an example of the idea. The more the task becomes complex, the more information is needed for training, the more reliable input data becomes difficult to identify.

To make system scalable we have implemented an architecture based on Web Services where each task is divided in subtasks. Each subtask is performed by a server which in turn will use other servers for implementing parts of the subtask. 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 in the CS department task will return all papers written by a person by accessing Citeseer. Another one will do the same on another digital library . The named entity recogniser server (whose role is to decide if a string is a name) will invoke these servers and integrate the evidence returned and decide if such evidence is enough to conclude that the candidate string represents a person.

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 [6]. Amilcare can be trained to work on rigid documents (e.g. Citeseer or Google output) by providing a handful of manually annotated examples, while it needs some hundreds of examples for more sophisticated cases [5]. All the servers are defined in a resource pool and can be inserted 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. In the CS website task, wrappers are defined for all the resources described in Section 3. The CS application works in the following way: a user submits a URL. The system returns a database populated with people's names, personal details, papers, projects, etc. 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 person name is missed by the system, it can be manually added by the user. The modules that receive as input the output of that name finder will then be re-run and

[2]http://www.aktors.org/technologies/Armadillo/

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).

3. THE COMPUTER SCIENCE DEPARTMENT TASK

The application used to describe the methodology is mining websites of Computer Science Departments. The goal is to discover who works in a specific department (name, position, home page, 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. The structure and layout of the web pages is different for every department but we can expect to find some common elements in Computer Science Departments web pages such as a people's page containing the list of people working in the department and many others.

3.0.1 Finding People Names

The goal of this task is to discover the names of the people who work in the specific department. This task is more complex than a generic Named Entity Recognition because many irrelevant people's names are cited in a site, e.g. names 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) such as Annie (www.gate.ac.uk) 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 NER tend to be quite slow if launched on large sites (e.g. the 1,600 pages of the CS department at the University of Southampton) 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 found. Then these seeds are used to bootstrap learning for finding further names.

3.0.2 Finding Seed Names

To find seed names, a number of weak strategies are combined that integrate information from different sources. First 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 home page (if any);
- **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 home page (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 home pages;

The information returned by the digital libraries (Citeseer and Unitrier) is used to confirm or deny the name identity of the string. If they return reasonable results for a specific name (i.e. not too few and not too many), this name is retained as potential name. Defining what a reasonable result for a digital library is crucial here. If a string is a valid name, a number of papers are returned, otherwise the output is either empty or with unlikely features. For example when querying Citeseer with the term "Smith" more than 10,000 papers are returned. This is the indication of a potential anomaly: the probability that a person writes more than 150 papers is quite low and the name can be discarded. Equally, when looking for a non-name (e.g. the words "Fortune Teller"), no papers are returned. We tend to use quite restrictive criteria for keeping reliability high (e.g. more than 5 papers and less than 50 returned by Citeseer); redundancy of information allows to bootstrap learning using just a limited amount of information, as already noted by Brin [2]. The results of the digital libraries are integrated with those of the classic Named Entity Recognizer run on a window of words around the candidate (so to avoid the problem of slow processing). At this point a number of names of people are available. They are in principle of three types:

1. correct (they are people working for this department);
2. wrong (they are not people: they are false positives);
3. people who do not work at this 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.

3.0.3 Learning Further Names

All the occurrences of seed names are then annotated on the site's documents. Learning is performed initially only on documents where a reasonable quantity of known names

are organized in HTML 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 elements (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. [3]. 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 [17].

3.0.4 Extracting Personal Data

To extract personal data (email address, telephone number, position, etc.) it is necessary to identify a dedicated web page (e.g. a personal web page). Again we combine information from Citeseer, HomepageSearch and Google to check if the person has a known page in the current web site. Otherwise we look for occurrences in the site in which the name is completely included in an hyperlink pointing internally to the site. It is then possible to extract personal data from the home page using a named entity recognizer (e.g. Annie) to easily identify them. In case some of the personal data are not found in the main page, subpages pointed by the home page are inspected. Only pages with an address under the same path are considered (e.g. www.aaa.edu/ãdomine/index.html and www.aaa.edu/ãdomine/contact.html refer to the same subdirectory.

3.0.5 Discovering Papers Citations

Discovering what papers are written by what members of the departmental staff is a very difficult task. It requires to identify the paper title and the authors, then relate 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 title of [7]) and cannot be 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. Nearly each department and each member of staff in CS departments provide a list of publications. Moreover papers are co-authored, so it is very possible that each paper is cited more than one time in 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 in different moments of times. 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 names, the seed examples are annotated and page-specific patterns are induced. We favour examples contained in HTML structures such as lists 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>
    Proceedings of the 13th International
    Conference on Knowledge Engineering and
    Knowledge Management (EKAW02), 1-4 October 2002
    - Sigenza (Spain), Lecture Notes in Artificial
    Intelligence 2473, Springer Verlag
    <br>
</li>
```

[3]Classifiers are induced in our implementation by Amilcare, http://nlp.shef.ac.uk/amilcare/, an adaptive IE system developed at Sheffield [6]

Simple wrappers relying on the HTML structure only would be ineffective, as there is no way to discriminate - for example - between authors and editors and title of paper and title of collection using the HTML structure only. More sophisticated wrapper induction systems are needed, as that provided by Amilcare [6], which exploits both XML structures and (para-)linguistic information [5]. Using a cycle of annotation/learning/annotation we are able to discover a large number of new papers. The cycle finishes when there is no more information to discover or the user decides to interrupt the cycle. Note that every time co-authorship among people is discovered in analysing the publication page of one specific authors, the paper is retained for annotation when the other names are considered (i.e. the redundancy is exploited again).

4. EVALUATION

The architecture mentioned is fully implemented and we are currently experimenting extensively on a number of CS web sites. We have experimented on the sites of (1) the Computer Science Department of the University of Sheffield (www.dcs.shef.ac.uk), (2) the Department of Engineering and Computer Science of the University of Southampton, UK, and (3) the Department of CS of the University of Aberdeen, UK. In the following we extensively report about the results on the site of the University of Sheffield. Results on the other sites are qualitatively largely equivalent.

4.1 Finding People's Names

Experiments show that names of people can be found with a high reliability: in the case of the Sheffield's department the system discovers 51 seed names of people belonging to the department as either academics, researchers or PhD students, 48 correct and 9 wrong. These names are used to the seed learning. Amilcare then discovers other 57 names, 48 correct, 6 wrong. This increases the overall recall from 37% to 74% with a very limited loss in precision (see Table 1). A qualitative evaluation of the errors is worth doing: the 9 false positives were: A. Schriffin, Eugenio Moggi, Peter Gray, Speech and Hearing, European Network, Department Of, Position Paper, The Network, To System. The first three are actually names of people; their recognition is due to the wrong identification of some web pages within the site as their home pages. They are all people for which Citeseer returns papers and they were all wrongly recognized at seeding time. The others (all recognized by the IE engine) are truly spurious hits for which a home page was also erroneously found. Their number is quite low in proportion to the total number of names. Recognizing these names as false hits is quite easy for a person, so if the results are to be used or checked by a person, they do not constitute a problem. One of them (Speech and Hearing) is actually the name of a group with its own home page. Concerning personal data, we just checked the identification of the correct home page; as people are accepted only if a home page is found, the quantitative results for people are also valid for home page identification.

4.2 Paper Discovery

As mentioned discovering papers is a very complex task. In our experiments a paper was considered correctly assigned to a person if it was a paper mentioned in the personal papers list of the author and the title is 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. For Sheffield, the seed procedure discovers 488 papers, the learning procedure returns 752. We did not attempt discovery of further papers for people with less than 3 examples. Checking their correctness is very labor intensive, therefore we randomly checked the papers extracted for 8 of the staff members for which the seed papers exceeded 6 examples; they are shown in Tables 2 and 3.

The use of IE increases significantly the overall recall rate which grows from 37 to 55 (+18%). Total recall is 47 for seeds and 66 for IE-based, precision 99.3 and 98.6 and F-measure 64.0 and 79.7 respectively (see Table 4).

5. CONCLUSION AND FUTURE WORK

In this paper we have proposed a methodology to extract information from large repositories (e.g. large Web sites) with minimum user intervention. 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. Experimental results show that the methodology can produce high quality results.

The user intervention is limited to provide 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 delete 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.

The idea of using the redundancy of information to bootstrap IE learning is not new, having been already proposed by Brin [2] and Mitchell [17]. The difference with our approach is the way in which learning is bootstrapped. Brin uses user-defined examples, while Mitchell uses generic patterns that work independently from the place at hand (e.g. the site or the page). We use both the above[4], but in addition, we exploit the redundancy of information and integrate information extracted from different sources with different

[4]Generic patterns are used in the named entity recognizer. User-defined examples are used to discover project names, a task not described in the paper.

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
seed discovery	129	51	48	3	0	94	37	51
IE-based discovery	129	105	96	9	33	91	74	87

Table 1: Results in Discovering People and Associated Home Page. The first line refers to accuracy in discovering names using the procedure for seed names (Citeseer+Google, etc.), the second one to the discovery using adaptive IE.

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
R1	33	14	14	0	19	100	42	59
R2	46	23	22	1	23	95	47	63
R3	12	7	7	0	5	100	58	73
R4	77	31	31	0	46	100	40	57
R5	32	15	15	0	17	100	46	63
R6	36	15	15	0	21	100	41	58
R7	84	47	47	0	37	100	56	71
Total	320	152	151	1	168	99	47	64

Table 2: Seed Paper Discovery Accuracy; each line represents the papers discovered for a person. Possible represents the number of papers present in the personal publication list page, Actual the number of papers returned by the systems. Actual results are divided in Correct, Wrong and Missing.

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
R1	33	20	20	0	13	100	60	75
R2	46	36	36	0	10	100	78	87
R3	12	11	11	0	1	100	91	95
R4	77	38	38	0	39	100	49	66
R5	32	25	25	0	7	100	78	87
R6	36	16	15	1	20	93	41	57
R7	100	22	19	3	78	86	19	31
R8	84	71	69	2	13	97	82	89
Total	420	239	233	6	181	97	55	70

Table 3: IE-based Paper Discovery Accuracy obtained by Amilcare using the seeds in table 2 to bootstrap learning.

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
seed	320	152	151	1	168	99.3	47.2	64.0
IE-based	320	217	214	3	103	98.6	66.9	79.7

Table 4: Paper Discovery: Grand total

levels of complexity. In this respect our approach is - to our knowledge - unique. As noted by Brin, great care is needed in order to select only reliable information for annotation for learning. The integration of different knowledge sources multiplies the available information, and therefore allows to seed learning only when multiple evidence is found. The experimental results described in this papers were obtained by fixing high precision (>90%) and testing the obtained recall. Other configurations are possible and the results could radically change. Further experiments are needed for verifying the behavior of the system in different configurations.

One relevant question for the effective usability of the methodology in real applications concerns the required level of accuracy (as a balance of precision and recall) the system has to provide. As Web applications are concerned, it is well known that high accuracy is not always required. Search engines are used every day by millions of people, even if their accuracy is far from ideal: further navigation is often required to find satisfying results, large portions of the Web are not indexed (the so called dark and invisible Webs), etc. Services like Citeseer, although incomplete, are a very successful. What really seems to matter is size: the ability to both retrieve information dispersed on the Web and create a critical mass or relatively reliable information. In this respect the proposed methodology is satisfying. Experiments show that in the case of paper discovery, it is able to discover a large part of the information that the digital libraries used to seed did not have (+50%). Precision is topping 90%, so the information provided is very reliable.

5.0.1 Challenges for IE

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.

5.0.2 Integrating Information from Different Sources

The proposed methodology is based on using the redundancy of information. Information is extracted from different sources (databases, digital libraries, documents, etc.), therefore the classic problems of integrating information arise. 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, 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. For example, a person name can be cited in different ways: N. Weaver, Nick Weaver and Nicholas Weaver are potential variation of the same name. But do they identify the same person as well? When large quantity of information is available (e.g. authors names in Citeseer) this becomes an important issue [1]. This problem intersects with that of intra- and inter-document coreference resolution well known in Natural Language Processing. We are currently focusing on mining websites, because this allows to apply some heuristics that very often solve these problems in a satisfying way. For example the probability that N. Weaver, Nick Weaver and Nicholas Weaver are not the same person in a specific CS website is very low and therefore it is possible to hypothesize coreference. Different is the case of ambiguity in the external resources (e.g. in the digital libraries). Here the problem is more pervasive. When querying with very common names (e.g. "John Smith") the results is quite disappointing, as papers by different people are mixed. This is not a problem in our approach because the information returned is used to annotate the site. Paper from people from other departments or universities will not introduce any annotations and therefore will not cause any problems. The same applies in case multiple home pages are returned: if they do not have an address local to the current site, the page is not used. In the generic case, though, this is a problem. We are currently using this strategy to recognize named entities from Reuters news and to find more information about a specific name. In this case we have to understand if the Ken Russell cited in a specific news is the famous regissuer or an MTI researcher. We are currently experimenting with a strategy that integrates also evidence from lexical chains extracted from generic ontologies. The idea is that an MTI researcher and a regisseur should produce different lexical chains (one concerning computers, the other concerning films).

Acknowledgements

This work was carried out within the AKT project (http://www.aktors.org), sponsored by the UK Engineering and Physical Sciences Research Council (grant GR/N15764 /01). AKT involves the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University. Its objectives are to develop advanced technologies for knowledge management. Thanks to David Guthrie for the contribution to the discussion.

6. REFERENCES

[1] H. Alani, S. Dasmahapatra, N. Gibbins, H. Glaser, S. Harris, Y. Kalfoglou, K. O'Hara, and N. Shadbolt. Managing reference: Ensuring referential integrity of ontologies for the semantic web. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02.* Springer Verlag, 2002.

[2] 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.

[3] F. Ciravegna, Alexiei Dingli, Daniela Petrelli, and Yorick Wilks. User-system cooperation in document annotation based on information extraction. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02*. Springer Verlag, 2002.

[4] 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.

[5] Fabio Ciravegna. $(LP)^2$, an adaptive algorithm for information extraction from web-related texts. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in conjunction with the 17th International Joint Conference on Artificial Intelligence*, 2001. Seattle, http://www.smi.ucd.ie/ATEM2001/.

[6] 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, Amsterdam, 2003.

[7] 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.

[8] D. Freitag and N. Kushmerick. Boosted wrapper induction. In R. Basili, F. Ciravegna, and R. Gaizauskas, editors, *ECAI2000 Workshop on Machine Learning for Information Extraction*, 2000. www.dcs.shef.ac.uk/ fabio/ecai-workshop.html.

[9] A. Masci P. Merialdo G. Mecca, P. Atzeni and G. Sindoni. From databases to web-bases: The araneus experience. In *Technical Report 34-1998*. Dipartimento di Informatica e Automazione, Universita' di Roma Tre, 1998.

[10] 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.

[11] S. Handschuh, S. Staab, and A. Maedche. CREAM — Creating relational metadata with a component-based, ontology driven framework. In *In Proceedings of K-Cap 2001*, Victoria, BC, Canada, October 2001.

[12] C. Knoblock, S. Minton, J.L. Ambite, N. Ashish, J. Margulis, J. Modi, I. Muslea, A. Philpot, and S. Tejada. The ariadne approach to web-based information integration. *International Journal of Cooperative Information Systems*, 2001.

[13] 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.

[14] 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.

[15] A. Keller M. Genesereth and O. Duschka. Infomaster: An information integration system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, AZ, 1997*, 1997.

[16] S. Miller, M. Crystal, H. Fox, L. Ramshaw, R. Schwartz, R. Stone, and R. Weischedel. Bbn: Description of the sift system as used for MUC7. In *Proceedings of the 7th Message Understanding Conference*, 1998. $www.itl.nist.gov/iaui/894.02/related_projects/muc/$.

[17] 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.

[18] I. Muslea, S. Minton, and C. Knoblock. Wrapper induction for semistructured web-based information sources. In *Proceedings of the Conference on Automated Learning and Discovery (CONALD), 1998.*, 1998.

[19] Maria Teresa Pazienza, editor. *Information Extraction: A multidisciplinary approach to an emerging information technology*. Springer Verlag, 1999.

[20] 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 *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02*. Springer Verlag, 2002.