

# Mining the Semantic Web: Requirements for Machine Learning

**Fabio Ciravegna,**

Department of Computer Science, University of Sheffield,  
Regent Court, 211 Portobello Street, Sheffield, S14DP, United Kingdom,  
F.Ciravegna@dcs.shef.ac.uk

## 1 Introduction

The availability of documents annotated using markups derived from ontological components is a prerequisite for the Semantic Web. A number of tools have been proposed for manual annotation of documents, e.g. (Staab et al., 2001); some of them use Adaptive Information Extraction to reduce the burden on the user side (Vargas-Vera et al., 2002). Relying on a manual process presents some risks for the Semantic Web, because it creates a bottleneck: convincing millions of users to annotate documents requires a world-wide action of uncertain outcome. Moreover there are some serious concerns about the quality of manual annotation either for user inability or for spamming (Dingli et al., 2003) (Dill et al., 2003).

To produce a viable and maintainable Semantic Web, large scale automatic annotation services, similar to today's search engines, are needed. They must be: (1) easily defined for a specific ontological component or service; (2) able to constantly reindex documents (so to solve problem of obsolete/misaligned annotation). SemTag (Dill et al., 2003) is a large scale annotation service whose goal is to annotate portions of documents using instances in a knowledge base. It is based on an architecture where documents are initially tokenized and preliminary annotations (URIs to instances or concepts in the ontology) are assigned to sequences of words and then they are disambiguated using sophisticated techniques. SemTag is included in an architecture (Seeker) that also supports querying the inserted annotations. SemTag is essentially a semantic search engine.

In this paper we describe an approach to annotation services based on information extraction and integration. We describe the methodology, a system that implements it, and identify requirements and challenges for future research in both Machine Learning and Text Mining.

## 2 Armadillo

Information can be available 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. Information can be extracted from different repositories with different reliability. Databases can be queried using an API or by defining or inducing a wrapper for their web front end (e.g. CGIs). When the information is contained in textual documents, extracting information requires more sophisticated IE methodologies based on linguistic analysis.

Information is often redundant, in the sense that multiple citations of the same information can be found in different contexts and in different superficial formats. When known information is present in different sources, multiple occurrences can be used to bootstrap recognizers that, when generalized, will retrieve other pieces of information, producing in turn more (generic) recognizers (Brin, 1998).

Armadillo is a tool for extracting and integrating information from large repositories (e.g. the Web). It learns how to extract information by using redundancy of information in the following way: (1) it mines a coherent portion of the repository (e.g. a web site or a class of sites) (2) it integrates information from different sources (e.g. digital libraries) and uses it to bootstrap learning from the repository (3) it discovers new information in the repository that in turn is used to bootstrap new learning until a stable information base is reached; (4) it stores the harvested information into a data base (a triple store) available on the web. The database can then be used either to access the extracted information or to produce indices for document retrieval.

There are different degrees of complexity in extracting from different sources. The more the complexity increases, the more the amount of data needed for training grows, and the extraction ac-

curacy can be lower: a wrapper for database output can be trained with a handful of examples and reach 100% accuracy, whereas full IE systems may require the annotation of corpora of millions of words and reach 70% accuracy. Considering that it is often possible to ask a user to annotate a handful of examples, but it is generally impossible to ask to annotate millions of words, Armadillo always starts learning from rigidly structured sources using examples provided by a user. Then it seeds learning on more complex sources (e.g. free texts) using the acquired information. Armadillo's architecture is generic and it is organized as a set of agents/semantic web services that can be composed to perform an application-specific task. Current applications include extracting information about Computer Science Departments (see below), painters and their work (names of painters and list of paintings and related images) and companies' contact addresses from their Web sites.

## 2.1 The CS Application

Consider the following application of mining websites of Computer Science Departments: (1) discovering who works in a specific department (name, position, home page, email address, telephone number) and (2) extracting for each person a list of published papers larger than the one provided by services such as Citeseer.

Discovering who works for a department is more complex than 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) 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. A two-step strategy is used in Armadillo: initially a short list of highly reliable seed names of people working in the department are found. Then these seeds are used to bootstrap learning for finding further 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: (1) digital libraries: Citeseer, the CS bibliography at Unitrier, Home-

PageSearch, (2) a generic Named Entity Recognizer: (Annie, [www.gate.ac.uk](http://www.gate.ac.uk)) (3) a search engine (Google). The information returned by the digital libraries is used to confirm or deny the name identity of the string in the CS world. If they return reasonable results for a specific name, this name is retained as potential name. A valid name will return a reasonable number of papers, otherwise the output is either empty (this is not a known researcher name) or with unlikely features (for "Smith" Citeseer returns more than 10,000 papers; it cannot be a single person). The results of the digital libraries are integrated with those of a classic Named Entity Recognizer run on a window of words around the candidate (so to speed up processing avoiding to apply the NER on the whole site). At this point a number of initial names are available that can be of three types: (1) correct (people working for this department); (2) wrong (terms that are not people's names); (3) people names mentioned in the site but do not work at this site (e.g. external coauthors of papers written by researchers of the department). Citeseer, Google and HomepageSearch are then used to look for a personal web page in the site. If such a page is not found, the names are discarded. 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. All the occurrences of seed names are then annotated on the site's documents and learning is started to discover other names. Learning is performed initially only on documents where a reasonable quantity of known names are organized in XML 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.). For example, every department has one or more pages listing their staff in some kinds of lists (personnel lists, tables supervisors/students, etc.). When some elements (e.g. four or five) are identified in a list or portions of a table, we train an IE system to recognize the known examples, using linguistic and/or formatting information. Applying the induced rules other names will be extracted from the same structure. Each time new examples are identified, the site is further annotated and more patterns can potentially be learnt. New

names can be crosschecked on the resources used to identify the seed list to gather further evidence that these are real names. In our experience, this is enough to discover a very large part of the staff of an average CS website with very limited noise (Prec:92, Rec:84). We are currently using combinations of the following evidence: (1) the name was recognized as seed; (2) the name is included in an XML 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. Detailed results of experiments are reported in (Dingli et al., 2003) A much more complex task is extracting a list of papers for each person more complete than the one provided by the digital libraries by mining personal bibliographical pages. Amilcare: (1) retrieves a list of papers from the digital libraries, (2) finds pages where some of those papers are mentioned (3) seeds learning and extracts new references. This task is far more complex than discovering names because the html structure of the lists does not help much. In one single list item, the authors' names and the title are inserted together with the editors and titles of collections where they were published, so more sophisticated IE methods are needed to discriminate among them. The full description of the application and methodology can be found in (Dingli et al., 2003).

### 3 Requirements and Challenges

The process described above poses some requirements and challenges on the technologies for learning to extract information.

#### 3.1 Different document types

From a natural language/Information Extraction point of view, it imposes requirements on the methodologies that can be used, in particular in terms of: (1) portability with no or limited user intervention (2) ability to cope with a variety of document types. When discovers papers citation in personal bibliographic pages, Armadillo learns to recognize regularities in the specific bibliographic style used: each person uses a different style and the task is discovering how the bibliographic page is organised. Here machine learning is essential: systems requiring manual rule development are not suitable because of the large number of pages to model. Regularities come both from the page layout (e.g. html lists when discovering new people names) or are language related (when discriminating among authors and editors in a bibliographic reference or in discovering departmental

membership in sentences like "Dr J. Smith has been promoted to professorship". This is typical strategy needed when copying with Web documents, as they can be very rigidly structured (e.g. when produced by a database) or fairly rigid (as manually compiled bibliographic pages that are regular, but with many inconsistencies) or free. Sometimes those types are mixed within a single document, which contains sections with different style. For example the main page of The Times (<http://www.timesonline.co.uk/>) contains free texts in the summaries of the articles, semi structured information in the headline lists (titles are mainly choppy sentences) and completely structured ones in the list of sections. The learning algorithm must adapt to the different document types smoothly without user intervention. Methods relying on deep linguistic analysis are bound to fail when confronted with rigid formatting (e.g. tables) and choppy sentences. Systems that rely on the use of formatting only (e.g. wrappers) are not able to cope with free texts and even choppy sentences. In Armadillo, learning is based on the (LP)<sup>2</sup> algorithm (Ciravegna, 2001) implemented in Amilcare (Ciravegna, 2003). It uses a methodology where the level of linguistic analysis is one of the parameters the learner tunes during learning; such tuning can be different for different pieces of information located in different parts of the documents (so to cope with mixed types).

#### 3.2 Kind of IE task

Another challenge for learning in Armadillo relies in the type of IE task to be performed. In principle fully-fledged IE is needed as the ontology can require complex event extraction. Moreover information about an entity is often dispersed in more than one document; therefore also multi-document IE is needed, including extracting information on events across multiple documents and multi-document coreference. Therefore the ML challenge for the IE methodology is to adapt to very complex tasks in a largely unsupervised way. Currently in Armadillo some very effective default strategies for multi-document extraction are used that drastically simplify the task; for example in the CS task, we expect that every potentially coreferent name (say J Smith and John Smith) are always coreferring, unless there is strong evidence from the IE point of view that they are not (e.g. they have different personal details). Although these strategies are effective in many cases, more sophisticated strategies need to be developed.

#### 3.3 Noise reduction

Noise is an intrinsic feature of the Web. Low quality annotation and spamming introduce noise, as

well as the Web dynamicity: pages are often built or modified in a careless way. As mentioned the format of personal bibliographic pages tend to be quite regular, but not totally: they tend to be modified every now and then when one item is added and the format used can change over time: for example the title of the paper that initially was presented in bold, can become in italic and the title of the collection the other way round, making using the format extremely difficult. ML methods able to cope with noisy data are necessary. For example, most of the wrapper induction methodologies require noise-free data. Amilcare is generally able to accept fairly noisy training data without degrading accuracy, but there is space for improvement in designing new algorithms.

Some noise can be introduced by the unsupervised method itself during the seeding and learning cycle. In our experiments in the CS task, three people seeds turned out to be wrong, i.e. they were people not working at the department. Keeping under control the effects of this kind of noise is a challenge. We currently use multiple evidence before accepting a seed at any stage in learning. It is a quite an effective strategy, and it filters most of the noise: precision in all the experiments always exceeded 90%.

### 3.4 Learning from sparse annotation

Most algorithms used for IE consider the set of annotations as positive examples and the rest of the document as negative examples. When seeding and learning in a page, though, this is not a suitable methodology, because many of the unannotated examples are actually information to be extracted. It is like having to learn from a corpus that is at the same time training and testing corpus. The strategy that we currently use is to limit the visibility window around annotated examples for each tag. Everything is in the window (e.g. +/- 5 words) is considered counterexample if it is not annotated. The idea is that the probability that an unannotated examples is included in such a limited window is very low. From an experimental point of view it works quite well, but more sophisticated methodologies able to use positive/negative/unknown examples are needed.

### 3.5 Page classification

Armadillo uses a number of strategies to identify documents where pieces of information can be present. For example it uses the titles of known papers as keywords for Google to identify personal bibliographic pages. In another application, Armadillo is required to extract the contact details of the companies to which the site belongs. In this case, it is necessary to identify the contact

pages in a site. What is needed is therefore a page classification strategy that works in (semi-) unsupervised way, by either (1) asking the user to provide a list of relevant pages and the system to cluster them and/or (2) discovering those pages automatically by using examples (e.g. triples companies/address/site).

## 4 Conclusion

In this paper we have outlined the challenges for Machine Learning models for mining the web for bootstrapping the Semantic Web. We have focused on the Armadillo methodology, but the requirements are actually relevant to a range of methodologies with similar aim. We will focus our future research activity on addressing them in a comprehensive way.

## References

- Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98*.
- Fabio Ciravegna. 2001. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI)*. Seattle.
- Fabio Ciravegna. 2003. 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.
- 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. 2003. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the World Wide Web Conference 2003*.
- Alexiei Dingli, Fabio Ciravegna, and Yorick Wilks. 2003. Automatic semantic annotation using unsupervised information extraction and integration. In *Proc. of the K-CAP 2003 Workshop on Knowledge Markup and Semantic Annotation*.
- S. Staab, A. Maedche, and S. Handschuh. 2001. An annotation framework for the Semantic Web. In *Proc. of the First Workshop on Multimedia Annotation, Tokyo*.
- M. Vargas-Vera, Enrico Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. 2002. 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.