Armadillo: Integrating Knowledge for the Semantic Web

Sam Chapman, Barry Norton, Fabio Ciravegna,
Department of Computer Science, University of Sheffield,
Regent Court, 211 Portobello Street, Sheffield, S14DP, United Kingdom,
{s.chapman, b.norton, f.ciravegna} @dcs.shef.ac.uk

1 Introduction

The Semantic Web, SW, needs semantically-based structured content to both enable better document retrieval and empower semantically-aware agents. One prerequisite for the SW is the widespread adoption of such structured knowledge, so without a universal acceptance other automated methods need to be employed to generate structured content from the existing unstructured web. Most of the current technologies available for creating structured content is based on static human centred annotation, very often completely manual, of documents.

Manual annotation is time-consuming and can introduce noise (Ciravegna et al., 2002), being incomplete or incorrect, hence decreasing the quality of the information. For these reasons, we believe that the SW needs automatic methods for annotating content. Automatic annotation services such as SemTag(Dill et al., 2003) and Armadillo(Dingli et al., 2003) intend to solve this problem by automatically providing SW content.

In this paper we describe the Armadillo approach to automatic annotation and detail the methods employed internally for integrating and ensuring consistency of elicited knowledge. Armadillo is a tool for extracting and integrating information from large repositories (e.g. the Web) developed at Sheffield. The methodology employed for validating and integrating the information is a series of weak evidential similarity tests, implemented through a library of String Metrics, detailed in section 2. Then the paper focuses on presenting the Armadillo tool and details a use case, relating the methodologies used.

2 SimMetrics: Similarity Library

SimMetrics is an open source extensible java library containing numerous Similarity Metrics[1].

A Similarity Metric is an algorithm that, given two inputs, typically strings, returns a measure of their similarity. Similarity measures come from a variety of disciplines, including statistics, DNA analysis, artificial intelligence, information retrieval, information integration and databases. They are usually simple algorithms e.g. Levenstein Distance, L2 Distance, Cosine Similarity, Jaccard Similarity and so on (a full list and descriptions of each method is beyond the scope of this paper and should be sought elsewhere[2]).

Similarity Metrics within the SimMetric library take as input two strings and return a float similarity measure ranging between 0.0 and 1.0, 0.0 being entirely different, 1.0 being identical.

This standardised approach facilitates combining these simple techniques to Integrate and manipulate knowledge within the Armadillo tool.

3 Armadillo

Information can be available in different formats on the Web: in documents, in repositories (e.g. databases or digital libraries), from agents, web services, web based API's, etc. Information can be extracted from different sources with differing reliability. When information is contained in textual documents, extracting it requires more sophisticated IE methodologies based on linguistic analysis and methods to ensure reliability of extracted information.

Information in repositories like the web is often redundant, in the sense that can be found in different contexts and in different superficial formats - the redundancy of information can be weak proof of its validity (Dingli et al., 2003). The contextual information around the extracted entities can be used as additional evidence: within Armadillo a statistical approach, based on SimMetrics, provides further weak evidence.

Armadillo learns how to best 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 RDF database. The database can then be used to access the extracted information (as detailed later) or to produce indices for document retrieval.

[1] http://sourceforge.net/projects/simmetrics/

[2] http://www.dcs.shef.ac.uk/~sam/stringmetrics.html

Armadillo typically starts learning from rigidly structured sources using examples provided by a wrapper, the user or previous data, then it seeds learning on more complex sources (e.g. free texts) using the acquired information.

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. Armadillo's architecture is domain independent and composable (Norton et al., 2004): agents/semantic web services can be composed to perform application-specific tasks like extracting information about Artists and their Artwork (names of painters and list of artworks and related images).

In order to explain in more depth the working of Armadillo an example is now detailed.

4 The Computer Science Department Application

Consider the following example task of mining websites of Computer Science Departments to find academics (name, position, home page, email address, telephone number and a list of publications more complete than the one provided by repositories such as Citeseer).

Simply discovering who works for a department is more complex than generic Named Entity Recognition (NER) as many irrelevant people's names are mentioned in a site, e.g. names of undergraduate students, secretaries, as well as names of researchers from external sites and hence irrelevant for this task.

Armadillo uses an statistical evidence based looping approach to aid the annotation/extraction task. Initially a quick list of potential names of people working in the department is found - this can be generated from a gazetteer, manual annotation, a wrapper or via simple crawling and NER. The set of initial potential data need not be considered a gold standard as more evidence is then sought for each potential academic.

Armadillo then loops through two repeating phases:

1. Evidence Building and Validation - relies upon building up a series of weak evidences to cumulatively rate knowledge for accuracy.
2. Extraction of potential knowledge - uses ML to learn from validated existing knowledge to find contextual patterns in rated data sources in order to elicit more potential knowledge which must then be validated.

4.1 Evidence Building and Validation

Evidence building comes from the series of weak approaches each of which are combined to give a rating which can be used to validate knowledge. Firstly additional redundant sources of the same data are identified via a simple search, e.g. google finding relevant URLs. This redundancy by itself is not valid evidence as the reliability of sources themselves must also be taken into account, e.g. a source with lots of potential academics is considered a better source than a source detailing just a single academic.

The rating of sources influences Armadillo's perceived validity of the potential academics found. Using the previously mentioned SimMetric library, section 2, the extracted entities are cross examined for simple string similarity: if something is suitably dissimilar from the rest, for example a failed capture, it is considered more possibly an error and its rating is decreased. At present the cross similarity tests are a combination of SimMetric's simple edit distance approaches, although more complex approaches could be employed. The context around a capture can also detail likelihood, e.g. the linguistic position within text, various other similarity techniques to provide further weak evidence could then be used, for example a vector space model of the page similarity, or a comparison of similarities in a link analysis of data sources.

This combination of multiple weak techniques can provide improved confidence in the extracted knowledge Armadillo finds.

4.2 Extraction of potential knowledge

Armadillo currently integrates Amilcare(Ciravegna and Wilks, 2003) $(LP)^2$ algorithm to extract potential new knowledge, but could be extended to encompass different ML algorithms, as for example T-Rex (Iria, 2005). The knowledge extraction is performed learning contextual rules on highly rated sources using the most likely instances as seed data.

This allows ML to use the contextual information of existing finds to suggest more potential entities which are investigated further using evidence to rate the found extractions, thus improving precision.

4.3 Trusted sources

Using Armadillos looping methods detailed above it quickly becomes apparent that the better and more reliable information sources, for example html lists on the desired site and external sources such as citeseer[3] and unitrier[4] are identified as oracles to quickly test new entities.

The combination of multiple sources and evidences facilitate extending and structuring data beyond the scope of existing web ontologies. This

[3] http://citeseer.ist.psu.edu/

[4] http://www.informatik.uni-trier.de/~ley/db/

generic evidence based approach can of course be extended to any domain where redundant evidence can be found.

5 Conclusion and Future Work

Armadillo shows that Integration of Information can be achieved by the combination of numerous weak evidential methods using contextual information and cross comparison. This method is employed to clean, normalise and even disambiguate data gathered via ML methodologies

Improved ML methodologies could benefit the efficiency of the process by targeting resources to the most appropriate and effective weak methods.

Future work will also add additional learning techniques to adapt to a given domain by identifying the better techniques to extract information, given a specific domain or task.

6 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), and the Dot.Kom project, sponsored by the EU IST asp part of Framework V (grant IST-2001-34038).

References

Fabio Ciravegna and Yorick Wilks. 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.

Fabio Ciravegna, Alexiei Dingli, Daniela Petrelli, and Yorick Wilks. 2002. 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.

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*.

Jose Iria. 2005. T-rex: A flexible relation extraction framework. In *Proceedings of the 8th Annual Colloquium for the UK Special Interest Group for Computational Linguistics (CLUK'05)*.

Barry Norton, Sam Chapman, and Fabio Ciravegna. 2004. Developing a service-oriented architecture to harvest information for the semantic web. In *First AKT Workshop on Semantic Web Services, (AKT-SWS04)*.