

Multi-strategy Definition of Annotation Services in Melita

Fabio Ciravegna, Alexiei Dingli, Jose' Iria, and Yorick Wilks

Department of Computer Science, University of Sheffield, Regent Court, 211
Portobello Street, Sheffield S1 4DP
{fabio|alexiei|jiria|yorick}@dcs.shef.ac.uk
<http://www.nlp.shef.ac.uk>

Abstract. The definition of methodologies for automatic ontology-based document annotation is a fundamental step in the Semantic Web vision. In the near future, semantic annotation services could become as important as search engines are today. Tools for the easy and effective development of such services are therefore needed. In this paper, we present Melita, a tool for the definition and development of ontology-based annotation services. Melita goes beyond the dichotomy rule learning Vs rule writing of classic annotation systems, as it allows adopting different strategies, from annotating examples in a corpus for training a learner to rule writing and even a mixture of them. It also supports users in defining and maintaining an ontology for annotation and in delivering the annotation service. The result is a tool easy to use and flexible to different user needs.

1 Introduction

The Semantic Web needs semantically-based document annotation to both enable better document retrieval and empower semantically-aware agents. Different annotation schemas are likely to be superimposed on a web page using different ontologies, reflecting different domains of interest over the same information as regarded by different actors. Most of the annotations are likely to be imposed by web actors other than the pages' owner, exactly like nowadays' search engines produce indexes without modifying the page code. In currently available technology, however, annotation is meant mainly to be statically associated to the document. Moreover, most of the available technology is based on human centered annotation, very often completely manual [9]. Manual annotation is difficult, time consuming and expensive [5]. Convincing users to annotate documents for the Web (e.g. using ontologies) is difficult and requires a worldwide action of uncertain outcome.

Static and manual annotation associated to a document is prone to:

- become obsolete, i.e. not aligned with pages' updates or evolving ontologies;
- be incomplete or incorrect with respect to a specific ontology, especially if the human annotator is not skilled enough;

- be irrelevant from the point of view of a specific ontology, e.g. a page in a pet shop web site may be annotated with shop-related annotations, but some users would rather prefer to find annotations related to animals.

Producing methodologies for automatic annotation of pages becomes therefore important. The initial annotation associated to the document (and any other static one) loses its importance because at any time it is possible to automatically (re)annotate the document. In the future Semantic Web, automatic Semantic Annotation Systems (SAS) are likely to become as important as indexing systems are for search engines nowadays.

Automatic annotation methodologies have been developed in the past in different research areas such as Information Extraction from text (IE)[15], wrapper induction [11] and machine learning [14]. Methodologies have been defined for:

- reducing the burden in some SW annotation tools using adaptive IE[16] [8] [5]
- crawling the Web in an unsupervised way for harvesting domain specific information [12] [14][4]
- produce generic annotation such as Named Entity Recognition [7].

These methodologies *per se* represent a partial solution to the problem. Annotation tools based on adaptive IE [16] [8] [5] require the manual (or semi-automatic) annotation of examples to train an underlying adaptive IE system. When the user thinks the IE engine has reached a satisfying accuracy, the annotation service can be released. These tools focus on sequential annotation of documents in a corpus. Goal of the annotation is to train the IE system, which in turn will generate resources (e.g. rules) for the annotation service. They are designed with naive users in mind, i.e. users not acquainted with IE: they allow to produce a SAS via document annotation only. Although many users are able to write satisfying pattern matching rules for many tasks, such tools do not provide any facilities for rule writing. Unfortunately, in some cases a number of texts need to be annotated until the system learns patterns users could summarize in a couple of minutes. In case of data sparseness (e.g. a type of information is rarely present in the texts), users must browse/annotate many documents before a number of relevant cases can be retrieved that is sufficient to produce a reliably trained system also for the sparse phenomena.

Fully or largely unsupervised systems (e.g. [14][4]) exploit the redundancy and/or regularity of corpora (or the web) to automatically produce annotations, using user feedback to retrain the system. User's contribution is generally limited to preliminary lists of names of relevant objects and the correction of final or intermediate results. From the point of view of the generated annotation service, they tend to work as a black box. Users are not generally expected to contribute in the development, for example by writing rules. The problem of data sparseness mentioned above does not apply because the approach is largely unsupervised. Nevertheless the opacity of the rule learner and the inability to operate on it can sometimes be irritating for some users [10].

Systems requiring manual development of rules (e.g. [7]) rely on the user's ability to generalize over examples and to capture regularities in the corpus. The current state of the art in IE shows that the average manual system outperforms machine learning based systems trained on the same amount of data [1]. This is mainly because humans can generalize better and more quickly than systems, i.e. they require less examples to produce a good annotation service. For example Josen et al [10] show how a human inspecting 50 examples of deeds of conveyance was able to slightly outperform an adaptive IE system trained on 200. When using a manual system, however, personnel trained in developing grammars are needed. It is not possible to enable naive users to develop applications and there is no way to learn from any pre-annotated resource that should be available. Moreover a corpus needs in any case to be manually annotated in order to test for the system accuracy. Such corpus is generally of more limited size than the one needed to train an automatic system.

In this paper, we describe a tool integrating two of the approaches mentioned above (rule writing and document annotation) in an integrated way. Users can either write patterns or annotate texts and run a learner (or both) according to their skills and momentary needs. Moreover, the tool supports users in a number of other steps in the definition of a SAS: from the definition/refinement of the ontology, to (assisted) document annotation, to the writing of annotation patterns to the delivery of annotation service.

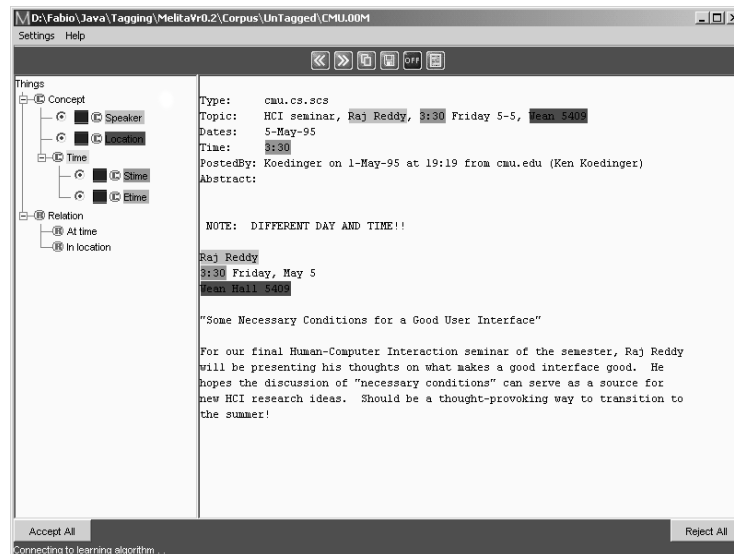


Fig. 1. The Melita Interface in the previous version: on the left the ontology is shown. On the right the document to be annotated is available. To annotate users select a concept and then use the mouse to highlight the relevant part of the document.

2 Melita

The current work extends Melita [5], an ontology-based text annotation tool. Melita’s main control panel is depicted in figure Figure 1. It is composed of two main areas:

- the ontology (left) representing the annotations that can be inserted; annotations are associated to concepts and relations. A specific color is associated to each node in the ontology (e.g. in Figure 1 the concept ”speaker” is depicted in gray);
- the document to be annotated (center-right). Selecting the the node in the ontology and then clicking on portion of text with the mouse inserts annotations. Inserted annotations are shown by turning the background of the annotated text portion to the color associated to the node in the hierarchy (e.g. the background of the portion of text representing a speaker becomes gray).

Melita provides support to annotation based on adaptive Information Extraction (IE). While the user annotates, an IE system (Amilcare [3]) monitors the annotations inserted by the user and - when similar cases are found in new documents - suggests annotations to the user.

The goal of Melita is to provide a way to produce annotation services using only knowledge of the domain. Melita was originally designed with naive users in mind. It does just require the annotation of documents using an intuitive interface, but it does not support users in an integrated way - it does not enable users to the manage IE rules or the ontology themselves, for instance. Melita supports annotation based on documents. Each document is annotated separately. There is no concept of corpus, apart from the set of documents already annotated that are used to train Amilcare. It is possible to browse documents from a corpus, but it is never possible to query the corpus in its entirety. In the next section we will describe how Melita was enhanced in order to overcome the limitations just mentioned.

3 The three focuses of interaction

A new version of Melita was designed and implemented that supports different tasks and interaction strategies for producing a SAS, from the definition/refinement of the annotation ontology, to (assisted) document annotation, to the writing of annotation patterns, to the delivery of the annotation service.

There are three focuses of interaction for the user:

- the ontology;
- the corpus, both as a whole and as a collection of single documents;
- the annotation pattern grammar(s), either user- or system-defined.

Users can move the focus and the methodology of interaction during the creation of the SAS in a seamless way, for example moving from a focus on document annotation, to rule writing, to ontology editing.

3.1 Initial Setup

Two objects are needed to start the definition of a new SAS in Melita: a corpus and an initial ontology. The ontology defines the labels to annotate documents in the corpus. The corpus will provide the material to train the underlying IE system or to help users in developing rules. The initial ontology can either be provided by the user or learned using (semi-)automatic methodologies [13] [2]. Melita can read ontologies in DAML+OIL, RDF and XI. In the average application this ontology generally represents an initial draft to be refined after exploring some of the documents in the corpus to be annotated.

3.2 Corpus Focus

The corpus provides the material to train the underlying IE system or to help users in developing rules. It also often motivates the refinement of the ontology. Melita provides facilities to access the corpus in three modes:

- browsing/exploring the set of documents;
- annotating single documents; documents are accessed in a sequential mode and annotated using labels from the ontology. The inserted annotations are passed to the learner to induce patterns; these will constitute the backbone of the first version of the annotation service. This modality is the classic provided by many annotation tools such as MnM [16], Ontomat [8] and the previous version of Melita.
- querying, to retrieve documents containing interesting information or exploring potential patterns in the corpus;

Here the focus is no longer only on the sequence of documents to annotate, but also the whole corpus can be queried and browsed as a whole. This kind of facilities is not included in any of the annotation tools we have seen so far. We will come back to querying in Section 3.4.

3.3 Ontology Focus

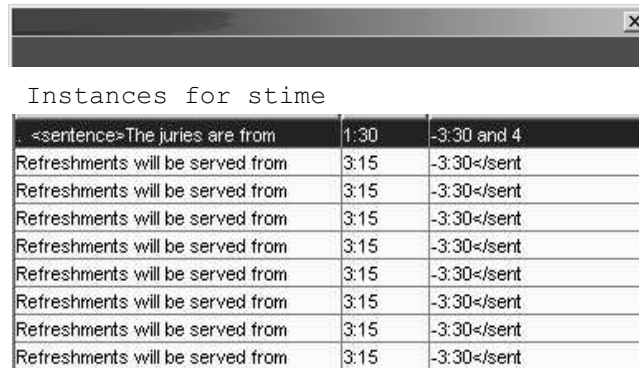
Focussing on the ontology is important to get a global view of the status of the provided annotation, for example by inspecting all the instances associated to a specific concept/relation that have been annotated in the corpus. This may be useful to:

- revise the ontology, e.g. decide to introduce a new concept or to split an existing one into two subconcepts;
- check the kind of phenomena or instances discovered so far in the corpus.
- understand the level of coverage of the current patterns, both induced and written (see next subsection);

During annotation, and especially at the beginning of the process, users may need to evolve the ontology. Users can for example realize that a specific concept should be actually split in two different subconcepts. Melita enables the user:

- to decide about changes to the ontology in an informed way: users can inspect all the instances of/annotation for a specific concept as they have been identified so far (or even inspecting some new ones retrieved querying the corpus), analyze the documents in which they appear (“the context of the information”) and decide for or against the possible modifications to the ontology.
- to rearrange the inserted annotations to the changes in the ontology in an efficient and effective way; for example if a concept is split all the annotations for that concept should be presented and the user should be enabled to change them according to the concept definition. The same applies to the patterns the users should have provided so far.

The focus on the ontology implies moving away again from the focus on single documents. The focus on the instances as annotated in the corpus as a whole (as opposed to annotated in a single document); see figure 2. Users are enabled to change the annotations without flipping through all the documents, but just focussing either on the instances themselves or to inspect just the portions of documents involved (not the whole documents) and eventually change the annotation associated to those instances.



Instances for stime

. <sentence>The juries are from	1:30	-3:30 and 4
Refreshments will be served from	3:15	-3:30</sent
Refreshments will be served from	3:15	-3:30</sent
Refreshments will be served from	3:15	-3:30</sent
Refreshments will be served from	3:15	-3:30</sent
Refreshments will be served from	3:15	-3:30</sent
Refreshments will be served from	3:15	-3:30</sent
Refreshments will be served from	3:15	-3:30</sent
Refreshments will be served from	3:15	-3:30</sent

Fig. 2. The list of instances associated to a concept. They are represented as they appear in the corpus (each line is an instance found in the corpus). It is possible to inspect the whole document in which the instance was found by clicking on the instance.

3.4 Grammar Focus

The grammar is the real goal of development in generating a SAS, being the resource enabling the final service. As mentioned, in the classic approach the IE system works as a black box, i.e. no rule modification is possible. Many users are keen to develop rules when they feel that this allows converging more rapidly to an optimal annotation service. This is because the average IE system may need to

see more examples than a person before converging to optimal rules. For example we noted that for recognizing a specific time expression the old Melita required the annotation of 2 or 3 dozens of examples in order to obtain 75% recall and 95% precision [6]. An average user will be able to derive very efficient patterns with a handful of examples, simply using their common sense knowledge about time expressions. Enabling users to write patterns (or to modify the induced ones) when they can/want can drastically reduce the time for producing the annotation service.

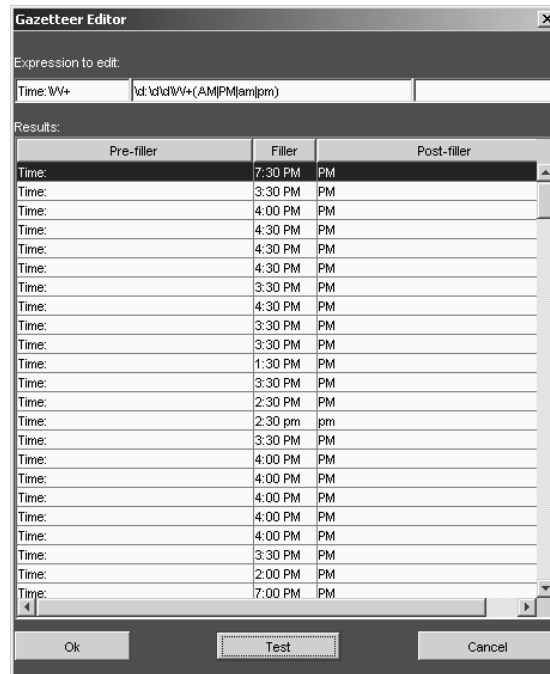


Fig. 3. The editor for regular expression showing matches for the patterns. A pattern is composed of a filler (center) and the contexts (left/right).

In Melita there are two types of patterns users can develop:

- classification patterns aimed at retrieving other pieces of information to reason on, either during IE pattern writing or during ontology revision; this feature is also useful in case of data sparseness in order to retrieve further documents to be annotated. These patterns tend to be generic patterns able to identify likely interesting information; they are not expected to be particularly accurate or to become as-is part of the annotation service.
- IE patterns aimed at incrementing the capabilities of the resulting annotation service, i.e. proper extraction patterns the user is contributing to the

IE learner. When a pattern is accepted, all the examples covered are automatically annotated in the corpus. IE patterns can either be compiled from scratch or be modified versions of induced patterns.

Accordingly, two types of editors are provided in Melita:

- a regular expression editor matching strings (mainly thought to be used for querying but that can be used for extraction patterns as well); this is an editor for users not particularly acquainted with Natural Language Processing (see Figure 3);
- an editor for patterns accessing the NLP features used by Amilcare; these features are derived by a linguistic preprocessor, e.g. Part of Speech Tagging, Gazetteer information and Named Entity Recognition, etc. The Amilcare’s preprocessor is based on Annie [7]).

Pattern writing requires to focus on the corpus as a whole: patterns need to be tested on the whole corpus (as opposed to single documents) to check their effectiveness. Users need to identify positive and negative examples covered by the patterns, either from the annotated documents or from the non-annotated ones. Facilities for testing patterns and presenting results are then needed. Such facilities are not present in any of the document-based annotation tools mentioned above. This is a modality that is typical of the tools that require rule writing.

3.5 Closing the Loop

The different views mentioned above tend to be quite separated in existing systems. For example in both the previous version of Melita and MnM [16] most of the functionalities mentioned above for document annotation are provided, but no facilities for modifying the ontology exist (for which other tools must be used) or to cope with the corpus as a whole (querying can be obtained using a search engine), and no rules can be edited (but Amilcare’s rule manager and editor can be used). The use of different tools makes very difficult switching among the different modalities and focuses.

In the new Melita, it is possible to move from the different views in a seamless way. When in corpus view, it is possible to access instances in the corpus and therefore moving to the view on the ontology. From the instances in the corpus is also possible to move to the patterns that recognize them (if any) and therefore moving to the focus on the grammar. When in grammar view, it is possible to move to the corpus view via the annotations inserted by the patterns in the corpus. Using the recognized instances it is possible to move to the ontology view. When in ontology view it is possible to inspect all the instances and how they are presented in the corpus (and therefore allowing to move on the corpus view) or the patterns used to recognize them (and therefore moving to the grammar view).

4 Conclusion and Future Work

Melita is a tool for defining and developing automatic ontology-based annotation services that provides different views over the task, based on three perspectives: the corpus to be used to develop the annotation service, the reference ontology and the patterns and grammars for annotation. In summary the following facilities are provided:

- Corpus:
 - sequential document annotation;
 - corpus querying (using patterns from the grammar);
 - from annotations to instances in the ontology (move to ontology view);
 - from annotation to the rules that inserted them (move to grammar view);
- Ontology:
 - ontology loading and editing (additions, modifications, etc.);
 - inspection of all the rules associated to an instance (move to grammar view);
 - inspection of how a (set of) instance(s) is presented in the corpus (move to corpus view);
- Grammar
 - grammar management (adding and removing rules)
 - rule editing;
 - rule testing and debugging;
 - inspection of all the instances associated to a rule (move to Corpus or Ontology view);

Melita allows to exploit the user abilities at their best. IE experts will mainly focus on developing rules, while non-IE experts will mainly annotate texts. Ontology engineers will use it to help validate the ontology. Average users will probably use a mixed strategy. We are organizing experiments to classify the behavior of different users and to quantify the gain provided by the different views with respect to the previous version. Details on experiments using the previous version can be found in [5] and [6].

Future developments will concern the inclusion of facilities from the Armadillo tool [4]. Armadillo is a system for unsupervised information extraction and integration from large repositories. Armadillo could be used to retrieve new documents from external sources (e.g. the Web) and apply/check the existing patterns. This could be useful for reducing the impact of data sparseness due to limitations in corpus size [4].

Melita is currently used by a UK company to generate an anonymization service for hospital patient records. The final service will discover, annotate and anonymize both the patient’s personal details and specific events that could allow identifying the patient. The cleaned records will then be made available for research in medicine. Melita uses as underlying IE system Amilcare [3], an adaptive IE tool specifically designed for document annotation that has been integrated also in MnM and SCREAM and it is currently under use in a dozen of industrial and academic sites.

Acknowledgements

This work is partially funded by 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.

The work is also partially funded by the IST-dot.kom project (<http://www.dot-kom.org>), sponsored by the European Commission as part of the framework V, (grant IST-2001-34038). dot.kom involves the University of Sheffield (UK), ITC-Irst (I), Ontoprise (D), the Open University (UK), Quinary (I) and the University of Karlsruhe (D). Its objectives are to develop Knowledge Management and Semantic Web methodologies based on Adaptive Information Extraction from Text.

References

1. Douglas E. Appelt and David Israel. Introduction to information extraction technology. IJCAI-99 Tutorial, August 2 1999. Stockholm, Sweden, "<http://www.w3.org/TR/1998/REC-xml-19980210>".
2. Christopher Brewster, Fabio Ciravegna, and Yorick Wilks. User-centred ontology learning for knowledge management. In *Proceedings of the 7th International Conference on Applications of Natural Language to Information Systems*, Stockholm, June 2001. Lecture Notes in Computer Sciences, Springer Verlag.
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, Amsterdam, 2003.
4. Fabio Ciravegna, Alexiei Dingli, David Guthrie, and Yorick Wilks. Integrating information to bootstrap information extraction from web sites. In *Proceedings of the IJCAI 2003 Workshop on Information Integration on the Web, workshop in conjunction with the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 2003. Acapulco, Mexico, August, 9-15.
5. Fabio 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.
6. Fabio Ciravegna, Alexiei Dingli, Yorick Wilks, and Daniela Petrelli. Using adaptive information extraction for effective human-centred document annotation. In I.Renz J.Franke, G.Nakhaeizadeh, editor, *Text Mining, Theoretical Aspects and Applications*, Lecture Notes in Computer Science. Springer Verlag, 2003.
7. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan". Gate: an architecture for development of robust hlt". In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, July" 2002.
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. 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.
10. M. Josen, P. Jongejan, G.J. Kersten, and E. Zopfi. Towards complexity measures: Guidelines for the feasibility of information extraction projects. In *Proceedings of the Dutch Conference on Artificial Intelligence*, 2001.
11. 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.
12. 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.
13. A. Maedche and S. Staab. Semi-automatic engineering of ontologies from text. In *Proceedings of the 12th Internal Conference on Software and Knowledge Engineering. Chicago, USA, July, 5-7, 2000*. KSI, 2000.
14. 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.
15. Maria Teresa Pazienza, editor. *Information Extraction: A multidisciplinary approach to an emerging information technology*. Springer Verlag, 1999.
16. 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.