

Semantic Web-based Document: Editing and Browsing in AktiveDoc

Vitaveska Lanfranchi¹, Fabio Ciravegna¹ and Daniela Petrelli²

¹ Department of Computer Science, University of Sheffield, Regent Court,
211 Portobello Street, S1 4DP, Sheffield, United Kingdom
{vita, fabio}@dcs.shef.ac.uk
<http://www.dcs.shef.ac.uk/>

² Department of Information Studies, University of Sheffield, Regent Court,
211 Portobello Street, S1 4DP, Sheffield, United Kingdom
d.petrelli@shef.ac.uk
<http://www.shef.ac.uk/~is/>

Abstract. This paper presents a tool for supporting sharing and reuse of knowledge in document creation (writing) and use (reading). Semantic Web technologies are used to support the production of ontology based annotations while the document is written. Free text annotations (comments) can be added to integrate the knowledge in the document. In addition the tool uses external services (e.g. a Semantic Web harvester) to propose relevant content to writing user, enabling easy knowledge reuse. Similar facilities are provided for readers when their task does not coincide with the author's one. The tool is specifically designed for Knowledge Management in organisations. In this paper we present and discuss how Semantic Web technologies are designed and integrated in the system.

1 Introduction

In the current form of the Web, content is designed and published for human readers and it is not typically tractable by machines; the Semantic Web, SW, is expected to make content processable in an automatic way via the addition of annotations. However, besides supporting automatic processing, the rich annotations behind the SW can improve the user's experience when dealing with documents and knowledge. Several methods of enriching Semantic Web documents have been proposed. One is to insert ontology-driven annotations that identify ontological instances in the document [8]. This type of annotation enables the capturing of document content, empowering better retrieval and reasoning.

A second method of enriching a document is to attach services: they can be associated to ontological instances and made available directly from the document in an automatic way [5]. Annotations and services can create a personalised view of the document, so that the reader can directly access its content (via ontology based annotation) and the additional information concerning it (i.e. the ontology and its knowledge base

can connect concepts in the document with external knowledge or documents - provided by the ontological-based services [7]).

A further way to enrich documents is to incorporate free text annotations in the form of comments [9]. They have become quite a standard feature¹, especially in the Knowledge Management (KM) world. Comments are used to integrate the text, adding information and knowledge not explicitly mentioned within the document: this is called *braindump*. For example, a lawyer could add explanations about referring to a specific regulation in a document (e.g. a EU directive), rather than others that could seem more relevant in the context of the document. In this case, comments are used for explaining the reasons that led to a specific formulation of the document itself, i.e. they are used to complement the knowledge in the document with knowledge about the process that generated it. A fundamental difference between braindump and ontology-based annotation is related to privacy. Typically, braindump is confined within an organisation's boundary as it contains the history, methodologies and motivations of a document; these are generally considered internal knowledge not to be shared with the outside world. As an example during the writing of this paper many comments were introduced by the authors as a way to discuss the paper content; however the form you are reading does not include it. The reader's braindump can comment not just the document itself, but even the author's annotations and comments. In this way it adds a further layer to the document knowledge.

All types of annotations, besides their different nature, share the same view of supporting "knowledge addition" by the different agents involved in the document lifecycle: e.g., the author(s) and the reader(s). Differences in the agent's role imply, in our view, a difference in management. Reader's tasks are different from the author's ones; for example, the ontology used for annotation can differ: inside an organisation a document may be written by the legal department using a legal ontology and accessed by the commercial department; the two departments are unlikely to share the same ontology.

In this paper, we propose to adapt and use SW technologies in order to support users during the lifecycle of a document, from production (writing), to consumption (reading) and maintenance (revision). By integrating the modalities of knowledge sharing offered by Semantic Web technologies, it is possible to create new opportunities for supporting users that can dramatically change the way documents are written and read. First and foremost, we claim that annotations (and especially ontological-based ones) must be generated as part of the document production step (i.e. editing) rather than during a separated step, as it happens in many of current approaches [8][11][5]. As a matter of fact, if annotations are added while documents are written, it is possible to use them to retrieve relevant content, moving from a situation in which they passively mark up the document content for future use (e.g. retrieval and reasoning), to one in which they contribute to reuse and sharing of knowledge during writing. This direction of research was preliminary explored by Carr et al [1] in the WickOffice editor, where knowledge about the domain of academics aided filling a pre-defined form (part A of an EPSRC project proposal). In that case, the knowledge used was static,

¹ Editorial tools like Microsoft Word and Adobe Acrobat provide tools to add comments.

i.e. its use was specified a priori by the application via the definition of the domain (knowledge about academics) and the form to be filled. The support was limited to filling pre-determined fields, while no additional knowledge was provided for writing the rest of the project proposal, especially its free text parts.

The system and the methodology we discuss here go one step further than that initial proposal. Conversely from the current technology our approach:

1. Supports all types of enrichment mentioned above (comments, ontologically-based and associated services, hyperlinking) both for author(s) and reader(s). Annotations can be added in layers, i.e. on top of other annotations.
2. Is able to automatically suggest ontological-based annotation so that annotations are immediately available and no separate annotation step is required.
3. Is able to monitor user actions while editing and to provide automatic suggestions about relevant content; support is not limited to filling forms and other pre-determined structures, but it is extended to free text as well; this enables timely reuse of existing knowledge when available.

The result is, in our view, a system that helps sharing and reusing existing knowledge. Its intended use is mainly for KM, in order to support sharing and reusing knowledge within an organisation. The system is called AktiveDoc and is discussed in the next section.

2 AktiveDoc

AktiveDoc is a system for supporting knowledge management in the process of document editing and reading. Its main feature is to support users (both readers and writers) in timely sharing and reusing relevant knowledge. In particular, document content and existing annotations are considered in the context of the user's previous knowledge; then further annotations and content are suggested for insertion. Proposals are gathered from different sources, i.e., from:

- *Information extraction processes applied on the document itself*: possible ontology-based annotations are automatically identified and proposed to the user. If accepted, they become part of the enriched document. Such annotations can be used again to connect to the KB and ontology as mentioned below.
- *Available structured knowledge*: existing annotations enable connections to knowledge bases and ontologies, so that part of the knowledge stored there can be proposed for inclusion in the document;
- *Querying the Web or other external repositories*, including querying both other documents in a repository and their annotations. Examples of tasks covered here are using a search engine or a SW harvester (e.g. Armadillo [2]) or retrieving pictures.

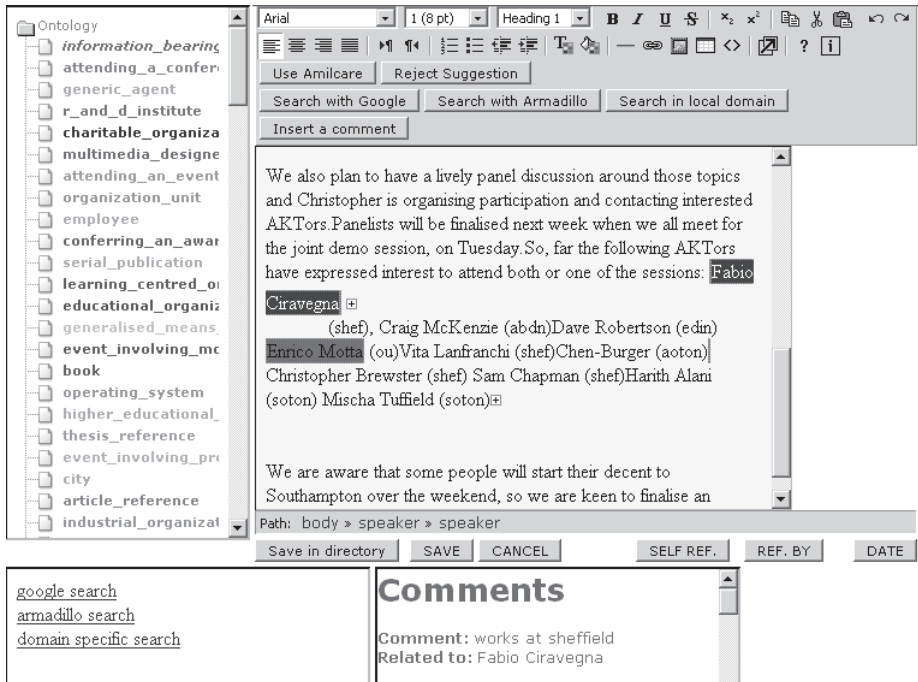


Figure 3- The ActiveDoc interface while comments are inserted: a “+” marks a comment (related to the concept Fabio Ciravegna); in the lower-right frame all the comments are listed.

Comments are organized in layers: a user may add comments to other users’ comments; they can also add annotations to their or other people’s comments. Finally, they can add System-suggested content (see next subsection) directly into comments. As for any type of annotation comments are stored in the database with authorship and level of confidentiality, to guarantee privacy and security.

2.4 Supporting content generation

As mentioned, one of the main features of ActiveDoc are active suggestions of relevant content to authors and readers in order to enable knowledge sharing and timely reuse. Knowledge is retrieved by external composable Web Services that exchange knowledge with the editor via the ontology.

When a portion of document is selected, services are made available depending on the annotations contained in the portion of text (if any) and on the string. For example, in Figure 3 in the lower-left frame the services associated to the annotated string “Fabio Ciravegna” are shown. Information about the individual are retrieved from a KB and other services are made available. Services are associated to types in the ontology and depend on the specific application.

A similar process was proposed in Magpie [5]. The difference with Magpie is that in

2.1 User Interface

The actual appearance of the editor depends on the application it is used for. In general, the interface is composed by (Figure 1):

- An editing window (top right) with formatting commands organized in toolbars;
- The ontology on a side panel (left);
- A set of lower frames that visualise system suggestions, contributions and proposed annotations (in Figure 1 the results of searching a proper name with a search engine, and a set of pictures from a database);
- Braindumps presented in a way similar to MS Word (shown in Figure 3).

The actual appearance for a specific application is decided during a setup phase where the different services are connected (mainly using Semantic Web Services) and assigned a portion of the editor for outputting results or receiving input. Input is provided by highlighting portion of text in the main pane and activating a service (e.g. search with a search engine).

2.2 Ontology-based enrichment

Concerning ontology based enrichment, the following kinds of services are provided: manual and semi-automatic annotation, and association of services. Manual annotations requires users to associate (portions of) documents to the ontology or KB in a way similar to what required by tools like Cream. Like in Melita [3], a graphical interface is provided where colours are associated to entities; a mouse click is needed to associate the selected text to a concept. The ontology used to annotate the document is chosen by the user uploading it in RDF format.

When working in a semi-automatic approach, the system learns from previous annotations how to suggest annotations while the document is edited. Again, the reference model is Melita's. Users can accept or reject annotations. User reactions to suggestions are used for further learning. The automatic detection provides efficiency (i.e. annotation is instantly available) while user corrections provide precision (i.e. only the truly important information is actually included in the annotation). Learning is based on a machine learning system connected to the interface. The current implementation connects to Amilcare [4]. The cycle annotation, correction and retrain is imported from Melita and the same strategy for avoiding intrusivity is used here [3]. The difference is that in the original implementation, Melita did not allow editing of document. This required implementing a specific annotation step. Moreover, it prevented the implementation of the strategies for content suggestion during editing mentioned below.

Learning To Harvest Information For The Semantic Web

 **Fabio Ciravegna**

Professor
University of Sheffield

 **Sam Chapman**

 **Alexiei Dingli**

 **Yorrick Wilks**

Copyright © 2004 f.ciravegna@dcs.shef.ac.uk

Abstract

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

[View Related Services](#) hide
[Google Search](#)
[View Related Item](#)
[View Personal Information](#)

[Author Related Information](#) hid
Name: Fabio
Surname: Ciravegna
Title: Dr
Mail: F.Ciravegna@Dcs.Shef.Ac.Uk
Telephone Number: +44011422219
Home Page:
[Http://www.Dcs.Shef.Ac.Uk/~Fabio](http://www.Dcs.Shef.Ac.Uk/~Fabio)

Figure 2- (part of the) the interface for browsing a document showing one suggestion for additional content. No editing facilities are provided, except for comments and adding annotations

2.3 Inserting unstructured annotations (Braindumps)

AktiveDoc accommodates insertion of free text comments into the document. As mentioned before, braindump can be done at any stage of the document lifecycle, both while editing and while reading. As in other tools, to insert a comment, the portion of document is highlighted and a button “Add Comment” is pressed. The comment will then be shown as a “plus” in the editor window (Figure 3). The comment can be expanded by clicking on the “plus”.

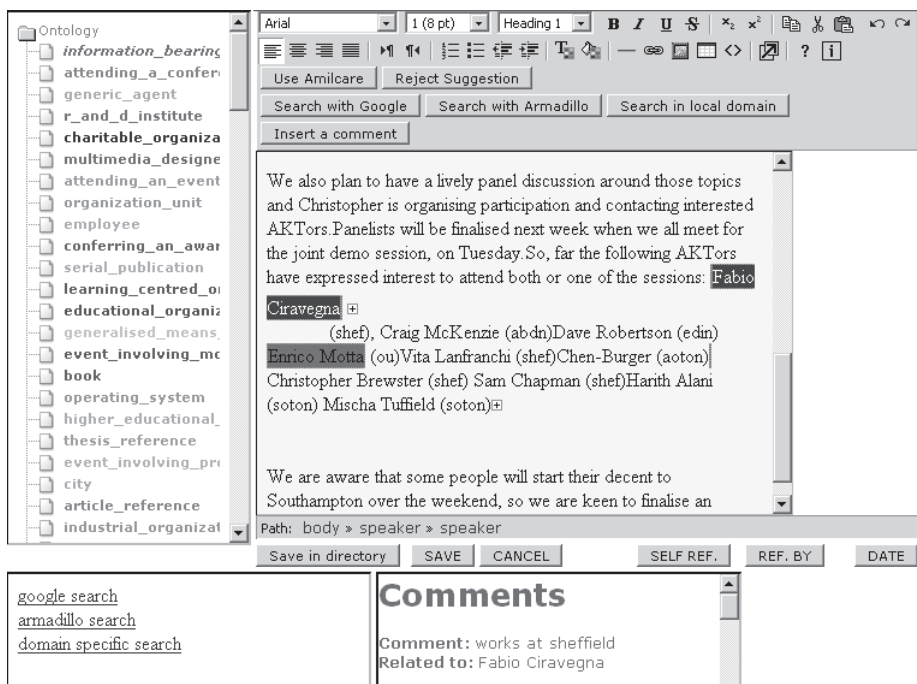


Figure 3- The ActiveDoc interface while comments are inserted: a “+” marks a comment (related to the concept Fabio Ciravegna); in the lower-right frame all the comments are listed.

Comments are organized in layers: a user may add comments to other users' comments; they can also add annotations to their or other people's comments. Finally, they can add System-suggested content (see next subsection) directly into comments. As for any type of annotation comments are stored in the database with authorship and level of confidentiality, to guarantee privacy and security.

2.4 Supporting content generation

As mentioned, one of the main features of ActiveDoc are active suggestions of relevant content to authors and readers in order to enable knowledge sharing and timely reuse. Knowledge is retrieved by external composable Web Services that exchange knowledge with the editor via the ontology.

When a portion of document is selected, services are made available depending on the annotations contained in the portion of text (if any) and on the string. For example, in Figure 3 in the lower-left frame the services associated to the annotated string “Fabio Ciravegna” are shown. Information about the individual are retrieved from a KB and other services are made available. Services are associated to types in the ontology and depend on the specific application.

A similar process was proposed in Magpie [5]. The difference with Magpie is that in Magpie annotation is not provided while editing but only in displaying the document.

As mentioned, providing services during editing enables retrieving new content to be added to the document, therefore knowledge sharing and reuse is possible for the author and not only for the reader. Also, in Magpie annotations are generated only using a named entity recognizer (eSpotter [12]), therefore they are quite shallow; moreover, a rule based Named Entity recognizer is used, and any addition of coverage to the recognizer requires rule writing by an expert. In AktiveDoc, annotations are either produced automatically by a system that learns from examples (Amilcare) or manually by the user. Also annotation is not limited to generic named entity recognition. In the current implementation connections to Armadillo (knowledge harvester) [2], Search engines (e.g. Google) and to structured resources (databases and knowledge bases [6]) are provided as shown in Figure 3 in the lower-left.

The activation of services is not automatic, but it requires a user action. This is done in order both to avoid spending CPU time on irrelevant tasks (that is one of the requirements for non intrusivity for automatic annotation [2]) and to avoid overwhelming users with (possibly irrelevant) suggestions (another requirement for automatic annotation).

Suggestions are presented to the user in a frame different from the one used for the document (and on which the user is working), so to avoid distracting the user attention when they appear. The interface currently allows both textual content (as in content retrieved from Google Web Service or Armadillo RDF repository) and multimedia content to be displayed (e.g. images retrieved by Armadillo see Figure 1).

Visualised suggestions can be inserted either directly into the document or as comment by dragging them in the wanted position.

3 Architecture and implementation

AktiveDoc is a client-server application integrated in a Web Based KM System. The system is based on an interface that interacts with user's actions and timely calls the appropriate modules for executing the actions. The main system's components are: (1) Annotation module and (2) Information module.

The information module (IM) is in charge of connecting to the appropriate information source and retrieving content to be suggested to the user; when the user selects a portion of document, the IM extracts the string and the contained annotations (if any) and sends them to the available services. Then it collects their results and presents it into the user interface. The annotations module is responsible for saving the annotations inserted by the user and for retrieving the automatic annotations provided by the system via Amilcare. Both modules are integrated in the user interface and are active only when a user's action requests them.

The system contains also an Ontology Module that is in charge of interpreting the RDF ontology the user loads and of visualizing it using appropriate style sheets. The editor interface is based on HTMLArea, a free, open source utility developed by interactivetools.com to convert a <textarea> field into a WYSIWYG editor. In this way HTML documents can easily be opened and visualized correctly in the system

and new documents can be created using the formatting facilities offered by HTML. Several frames are inserted in the main interface to allow displaying the ontology, the available services and the retrieved content. The interface is based on Javascript and JSP functions. Documents and annotations are saved in a MySQL database. Support for Semantic Web Services is provided by the Armadillo infrastructure [10].

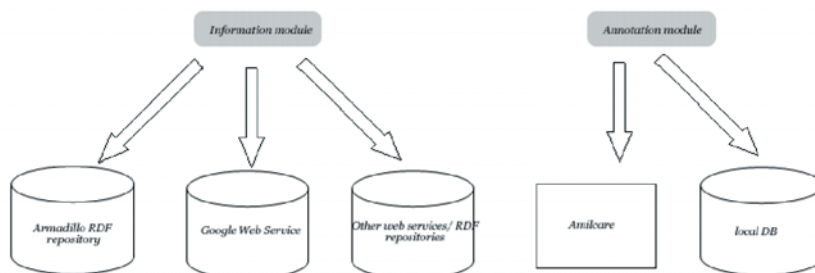


Figure 5 – An example of AktiveDoc application architecture

4 Discussion and conclusions

In this paper we have presented a tool for supporting users during the lifecycle of a document, from production (writing) to consumption (reading and publishing). The tool integrates Semantic Web technologies to support users in adding knowledge to the document. In particular, writers can receive suggestions on relevant content (enabling reuse of knowledge) and can be supported in producing ontology-based annotations (that will empower the Semantic Web). Also textual comments are enabled in order to add further knowledge to documents. Potentially³ the system is also able to suggest relevant content after the document is published, in order to allow document maintenance (e.g. newly available content could be communicated to the author even after the release of the paper).

Readers are able to access the enriched document produced by the authors using AktiveDoc. Moreover, in case their task differs from the author's one (e.g. they use a different ontology), they receive the same kind of support authors receive (relevant content suggestion, ontology based annotation, etc.); this enables reading in the context of their knowledge rather than in the author's one.

AktiveDoc has been designed mainly with Knowledge Management in mind, specifically in order to help reusing and sharing knowledge. These are fundamental needs in enterprises: according to some recent statistics, knowledge workers spend between 15% and 35% of their time in searching for knowledge⁴. Also, "lack of efficient publishing capabilities for digital content costs organizations \$750 billion annu-

³ This is a potential feature because it has not been implemented yet.

⁴ KMWorld Volume 13, Issue 3, March 2004

ally due to wasted time spent by knowledge workers seeking and capturing information necessary for them to do their jobs⁵⁷. By providing both ontology-based annotations and the suggestion of relevant content, we enable knowledge reusing. Knowledge sharing is empowered by layered comments and also by the searching capabilities provided by ontology-based annotations.

Annotations and services are stamped with authorship and are not saved in the document, so to allow confidentiality when needed. Marking authorship has also two other positive side effects. On the one hand it can contribute to identifying experts, a well known problem in large organizations. Associating annotations to documents means having coped with a problem, therefore it is possible to identify who works on specific problems by inspecting what documents a person has worked on. In traditional environments only the author can be tracked, not the readers. On the other hand, it allows implementing strategies of company management that rewards who shares knowledge within the company. The amount of sharing can be counted starting from the value and quantity of annotations provided to documents.

Future work on AktiveDoc will include the extension of the base of services provided and a field user test in a KM environment.

References

1. L. Carr, T. Miles-Board, A. Woukeu, G. Wills and W. Hall. The Case for Explicit Knowledge in Documents. In Proceedings of ACM Symposium on Document Engineering , pages pp. 90-98, Milwaukee, Wisconsin.
2. Fabio Ciravegna, Sam Chapman, Alexiei Dingli, Yorick Wilks. Learning to Harvest Information for the Semantic Web. In Proceedings of the 1st European Semantic Web Symposium, Heraklion, Greece, May 10-12, 2004
3. Fabio Ciravegna, Alexiei Dingli, Daniela Petrelli and Yorick Wilks. User-System Cooperation in Document Annotation based on Information Extraction. In Asuncion Gomez-Perez, V. Richard Benjamins (eds.): Knowledge Engineering and Knowledge Management (Ontologies and the Semantic Web), Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), 1-4 October 2002 - Sigüenza (Spain), Lecture Notes in Artificial Intelligence 2473, Springer Verlag
4. Fabio Ciravegna and Yorick Wilks: Designing Adaptive Information Extraction for the Semantic Web in Amilcare, in S. Handschuh and S. Staab (eds), Annotation for the Semantic Web, in the Series Frontiers in Artificial Intelligence and Applications by IOS Press, Amsterdam, 2003.
5. M. Dzbor, J. Domingue and E. Motta: Magpie: Towards a Semantic Web Browser. In Proc. of the 2nd Intl. Semantic Web Conf. (ISWC). 2003. Florida, USA.
6. Hugh Glaser, Harith Alani, Les Carr, Sam Chapman, Fabio Ciravegna, Alexiei Dingli, Nicholas Gibbins, Stephen Harris, M.C. Schraefel, and Nigel Shadbolt CS AKTiveSpace: Building a Semantic Web Application. In Proceedings of the 1st European Semantic Web Symposium, Heraklion, Greece, May 10-12, 2004

⁵ A.T. Kearney, Network Publishing study, April 2001

7. C. A.Goble, S. Bechhofer, L. Carr, D. De Roure, W. Hall. Conceptual Open Hypermedia = The Semantic Web?. In Proc. Of The Second International Workshop on the Semantic Web, Hong Kong, 2001.
8. S. Handschuh, S. Staab. CREAM - CREATing Metadata for the Semantic Web. Computer Networks. 42, pp. 579-598, Elsevier 2003
9. J. Kahan, M. Koivunen, E. Prud'Hommeaux, and R. Swick. Annotea: An Open RDF Infrastructure for Shared Web Annotations. In Proc. of the WWW10 International Conference. Hong Kong, 2001.
10. Barry Norton, Sam Chapman and Fabio Ciravegna. Developing a Service-Oriented Architecture to Harvest Information for the Semantic Web. In Proc. Of 1st AKT Workshop on Semantic Web Services, 2004.
11. Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt and Fabio Ciravegna "MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup", The 13th International Conference on Knowledge Engineering and Management (EKAW 2002), ed Gomez-Perez, A., Springer Verlag, 2002.
12. Jianhan Zhu, Victoria Uren, and Enrico Motta. ESpotter: Adaptive Named Entity Recognition for Web Browsing. To appear in Proc. of Workshop on IT Tools for Knowledge Management Systems at WM2005 Conference, Kaiserslautern, Germany, April 11-13, 2005.