Creating semantically enhanced documents with the help of text mining

Mihály Héder, Domonkos Tikk

Department of Telecommunication & Media Informatics
Budapest University of Technology and Economics
H-1117 Budapest, Magyar Tudósok körútja 2., Hungary
`mihaly.heder@computer.org, tikk@tmit.bme.hu`

Abstract

As users keep devising new and more creative ways to use the Web, the amount of textual content is enormous and ever growing. But the basic structure of a Web document remained unchanged: a mixture of text and markup. When creating a document, the user rarely has the opportunity of embedding semantic information in the content because editor applications do not have such a feature or it is too difficult to use. We think that the creation of semantically rich documents is the best facilitated by a neat content editor with text mining technology running in the background. In this paper we propose a system named Docuphet that meets these requirements.

1 Introduction

The success of the Wikipedia project illustrates the tremendous potential of the everyday Web user for creating vast amount of content. Other content-creation projects hold control over the submitted material by a rigorous review process or by a delegated editorial staff. These initiatives has never been able to produce the same amount of content. The NuPedia project which was started before WikiPedia is now only accessible in the Internet Archive. Citizendium, an other example of controlled encyclopedias has only tens of thousands articles while WikiPedia has millions.

Of course the quality of these articles is a subject of debate. The content representation however, is similar in the majority of cases. Almost every traditional encyclopedia-like content repository uses some form of page formatting markup and plain text. Some of them offer a limited semantic vocabulary to categorize the article, mark the date, creator, and some keywords.

There are research projects aiming to capture not only the formatting but also the semantics of the text while editing the content. The majority of these applications define themselves as “semantic wikis”. These applications offer functionality for embedding semantic expressions (usually RDF triples) by hand. That is, the user has to explicitly define the *property* and the *value* of the semantic expression involved, using a special markup. This requires certain special skills from the user and has a negative impact on the size of the potential audience.

An other approach is to annotate the text off-line, after the content was created, without human supervision. To achieve this Natural Language Processing (NLP) technology, namely Information Extraction (IE) is required. Considering the huge amount of textual data already present on the Web, these researches are very important.

Undoubtedly, it would be fruitful to provide tools for the everyday Web user to ease creation of semantic annotations while editing web content. The Docuphet [6] project aims at discovering and experimenting with the possible solutions of the problem.

2 Project goals and system overview

The main goal of Docuphet is providing a system which enables the user to create semantic annotation easily. This is carried out by a tool with intuitive user interface and intelligent text processing support running in the background. While the user types, IE applications are running on the text already available. Then, based on the extracted information, textual statements are formulated and presented to the user. If the user confirms the validity of the statements, the system embeds semantic annotations into the text, without requiring tiresome work and expert knowledge.

We don't rely on any special knowledge or experience of the users, but they may has knowledge about the particular content they are currently editing. This way they could be able to decide whether a given textual statement about the content is correct or not. In Docuphet, semantic annotations are inserted into the document by this manner.

Another property of the system is that it stores text and annotation together. This integration has many advantages. For instance when the text changes no extra operations are needed to look up the corresponding semantic annotations. Moreover, if the annotations were store separated, maintaining of extra identifiers would be necessary to link text and annotation.

To make it accessible and runnable without installation the client part of the system runs in an ordinary web browser.

It is a natural requirement against every system to support multiple languages. This requirement was considered when developing every component of the system. However, many IE techniques are language specific. Therefore a primary language was defined, the Hungarian.

The main point in creating semantically annotated text is enabling applications to search in, retrieve, and process the content in a more intelligent way. With a proper ontology, even machine reasoning becomes possible. However, to keep the project focused, these subjects are currently out of the scope. Docuphet concentrates on the creation of annotations only.

Currently Docuphet is a research and experiment project aiming for evaluating different solutions. Therefore in the case of some components, extra development iterations would be necessary to make them efficient under heavy load.

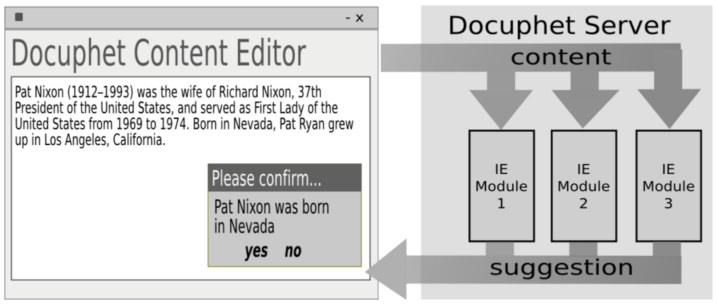
Docuphet Content Editor
Pat Nixon (1912–1993) was the wife of Richard Nixon, 37th President of the United States, and served as First Lady of the United States from 1969 to 1974. Born in Nevada, Pat Ryan grew up in Los Angeles, California.
Please confirm...
Pat Nixon was born in Nevada
yes no
Docuphet Server
content
IE Module 1
IE Module 2
IE Module 3
suggestion

Figure 1: The overview of the components and the data flow of the Docuphet system.

2.1 The main components and activities of Docuphet

The user interacts with the Docuphet Content Editor's web interface. The web application forwards the created content to the server via AJAX calls. On the server the content is distributed to various IE modules. The modules may compose one or more *annotation suggestion*s. Each suggestion comprises a textual statement or question, a numeric confidence level, and one or more semantic annotations (one per each positive answer). The confidence level is a real number of range between 0 and 1, which gives the certainty about the validity of the suggestion. Under a certain (configurable) level the suggestion is automatically dropped. The module's suggestions are collected on the server and are sent back to the client, where pop-up questions are presented to the user. If the user confirms the statement the system inserts the annotation into the content. When the user finishes editing, the full annotated document is sent back to the server and saved there. Figure 1 provides an overview of the system.

In the next section we discuss some key issues of the Docuphet components: editing the content, annotation storage, text processing.

3 Technology overview

3.1 Editing content

There is an excess of tools for creating content on a computer. These can be categorized in many ways. One aspect is the mode of editing. There are WYSiWYG editors like desktop word processors. Other, structured editors have two views: one for editing and an other for viewing the documents. When using an editor from the latter category, the user handles objects like “section”, “title”, “paragraph”. This category includes wiki editors, publishing tools, docbook [5] editors and scientific editors for LaTeX . In these applications the final formatting is done with style class files or style sheets.

Another aspect is the technology used to implement the editor. There are two main categories: the more function-rich desktop applications which need installation and the web-based editors [8, 23] which require only a web browser to run. Editors in the latter category has been traditionally simpler but as AJAX become widespread the complexity of these applications is almost equals to the desktop ones.

The special requirements against the editor used in Docuphet led to developing a completely new solution instead of heavily modifying an existing one. The Docuphet Content Editor (DCE) is an easy to use "What You See is What You Need" editor, which is capable of editing the structure of a document without the need of learning a markup language. DCE also handles the communication with the server, the representation of suggestions and it is capable of inserting the annotations.

3.2 Content storage

For selecting the best of the plethora of content storing formats we set up the following requirements:

1. It should be simple, to ease the implementation of a web editor for it.
2. It should be standardized, stable and free to use.
3. Proper support (documentation, examples, templates, editors, tools) is desired.
4. It should be extensible to carry annotations
5. The minimum functionality: paragraphs, sections and titles, lists, program listings, emphasizes, images, tables, links.

Many options were evaluated: RTF, texinfo, troff, wikitext, XHTML, Docbook, DITA, $\LaTeX$, ODF.

Because of the variety of convenient XML processing tools we dropped the non-XML formats. We also dropped ODF because of its high complexity. DITA and CDF (WCID) are too specific for our purposes. From the remaining two candidates we decided in favor of DocBook, because this format is purely structural, it doesn't contain any markup relating to the formatting of the documents. Moreover, it's grammar is defined in an easy-to-subtype Relax NG format, and it is easy to extend to carry semantic annotations.

3.3 Storing semantic annotations in the content

Many possible technologies were evaluated for storing semantic annotations in a DocBook document: HTML Metadata, RDF XML, GRDDL [9], Microformats and RDFa [26]. Finally we have selected RDFa because of its many advantages.

The RDFa [26] has been developed by W3C and is currently in the status of working draft. RDFa offers a technique to turn an arbitrary part of an XML document into an RDF triple. The technology is primarily aimed at annotating XHTML documents but also capable of handling XML documents from other namespaces.

RDFa was favored because it is easy to integrate with other XML formats, like Docbook. RDFa allows to annotate every part of a document, while it is still relatively easy to retrieve the RDF triples from the XML. To sum up, in Docuphet we use a special DocBook profile, which we call DocBook/RDFa.

3.4 Extracting semantic information from the text

The several Information Extraction (IE) techniques are aiming at getting structured data from an ustructured text.

3.4.1 Named Entity Recognition

A typical IE task is to recognize named entities in the text (called Named Entity Recognition or NER). A named entity is a natural textual identifer of an object. Examples of named entities are person names, names of companies, locations, names of products, addresses, telephone numbers, email addresses [20].

The difficulty of recognizing a named entity depends on its type. Telephone numbers, e-mail addresses can be easily recognized with simple regular expressions. Recognition of personal names and locations is more difficult but it can be effectively supported with extensive list of known instances. The recognition of a company or product name can be much harder, because these can be created without any constraint. In such cases certain features can be analyzed, e.g. surface clues (capitalization, numbers, special symbols), the frequency of the candidate named entity, the position of the candidate entity in the sentence or in the whole text. Grammatical and morphological analysis may also help.

The Docuphet framework contains a general purpose named entity recognizer engine, the JNER, implemented in java. This component comprises several modules, all of them analyzing the text with a different technique. Many of them use catalogs, for instance catalog of given names, company suffixes, locations. Others are based on regular expressions. It is possible to configure external tools, a morphologic analyzer or a stemmer, for instance. Other external tools can serve as connector to a database (i.e IMDB, DMOZ), or to a search engine (google, wikia search).

Although usually not considered as named entities, denomination of professions (like painter, composer, engineer) and human property (blond, tall) recognition are also carried out by JNER.

Conforming with the philosophy of Docuphet (ask relevant questions from the user), in this system named entity recognition can also be done by asking decisive questions about the named entity candidates from the user.

3.4.2 Information Frame Recognition

Another IE technique is Information Frame Recognition (IFR). We define an Information Frame (IF) as an RDF triple in which at least one value of the three is missing and thus substituted by a variable name. The class of the missing component(s) may be known. An example IF:

```
<X (a person)>,<location of birth>,<Y (a location)>
```

In our definition IFR means the recognition of instances of an IF in the text by identifying the missing components of the triple. For instance in the sentence "Pat Nixon was born in Nevada in 1912", we can recognize an instance of the IF detailed above:

```
<Pat Nixon>,<location of birt>,<Nevada>
```

Evidently, to solve an IFR problem it is very helpful to have the named entities and their types identified beforehand. In some cases other rules may be appropriate, i.e. stating that the elements of an IF must be present in the same paragraph or sentence. While limiting the number of potentially identifiable information frames, this constraint has many practical advantages: this way it is possible to start the IFR process before the whole content is available, and there is a significant reduction in computation time.

JFrame is the IFR component of the Docuphet framework, written in java. In JFrame information frames can be defined and the corresponding recognition rules can be given. The input of the module is a token stream in which the named entities and their types are already marked. A JFrame IF definition may have rules about the class of named entities in the token stream, the lemmas in the token stream and their order. Custom rules can be written in java. Conforming with the way Docuphet works as described in Section 2.1, JFrame also provides a confidence level for every recognized IF instance.

3.4.3 Sentence recognition

To support the recognition of information frames, a sentence recognizer was developed, named to JSentence. The component implements a modified version of the algorithm described in [22]. The Hungarian configuration of the tool was tested on the Szeged2 corpus [21], and it achieved a remarkable 98.54 % precision on the 82096 sentences of the corpus.

3.4.4 Classification

A category label of a given text is a valuable information on its topic. Knowing the category helps not only to find the document but also to select the relevant questions about the content and drop other unrelated ones. The Docuphet framework is prepared to collaborate with the hitec3 categorizing system [11].

4 Application examples

In this section two demo scenarios are presented for Docuphet. BioBase is a web site to collect biographies, similar to the ones in Magyar Életrajzi Lexikon (Hungarian Biography Encyclopedia) [12], FlatBase is a flat advertisement portal. In the case of the biographies, Docuphet is configured to recognize IFs based only on the entered text, while with the FlatBase portal the entered text is analyzed first, but then, if relevant information is still missing (e.g. floor no.) Docuphet creates automatically questions, to fill up the advertisement database properly. Both applications are configured to work on Hungarian text.

The main activities in both scenarios are the same as described in Section 2.1. Except the style sheets, the Docuphet Content Editor and the Document Server component versions are identical, the differences are in the way how annotations are produced. Therefore, in the following sections only this part is detailed.

4.1 BioBase

In BioBase a two layered IF recognition is implemented. In the first layer NEs are identified as follows:

1. JNER analyzes the text configured with all the NER rules available for Hungarian language. Currently this includes Person Name recognition (male and female distinguished) based on regular expressions and given name catalogs, Postal Location recognition based on catalogs, email and telephone number recognition based on regular expressions, profession name recognition based on a catalog, date recognition based on a custom java component, regular expressions and a catalog of months and a nationality recognizer based on catalogs.

2. JNER assigns confidence level to every recognized NE. Over confidence level 0.95, an annotation suggestion is created with the following RDF triple:

   ```
   <article id>,<related to [NE type]>,<[NE value]>
   ```

   where *NE type* and *NE value* are substituted with the actual values. The annotation suggestions also have level of confidence. In this case this is set to 0.95. The Content Editor accepts every annotation above 0.9 without asking a confirming question from the user, and to avoid frustration on the user's part.[1] The *target* of the annotations (the location where the annotation is placed) is the node before the given paragraph. From these annotations, a "tag cloud" can be generated with typed tags (Names, Locations, etc.) after saving the article.

3. For the top three Person and Location NEs with confidence level between 0.8 and 0.95 an annotation suggestion of value 0.8 is created with the question "This article is related to the Person/Location (Value)" with the same target and RDF triple as in the previous step.

Based on the NEs recognized, the following Information Extraction attempts are made:

1. First, Person who the article is about is identified. This is done by creating suggestions on the first Person NE found (in the title or in the text) with the triple

[1] Every annotation can be easily removed by the user by deletion or with the undo action.

```
<article id>, <describes the life of>, <[NE value]>
```

where the target is the beginning of the article, the confidence level is 0.8. This is repeated with the subsequent Person NEs until a positive answer is given by the user. (In our experience the article is nearly always about the first Person mentioned).

2. If the subject of the article is known, date and place of birth and death are attempted to be identified next. This is done by analyzing the Date and Location tokens in range of 7 tokens around every occurrence of the subject. Extra confidence is added if the past tense forms of lemma “született“ (was born), “elhúnyt—meghalt“ (died) are present around a particular Date or Location (these are often omitted however).

3. Nationality, profession and parent’s name are identified in a similar way as described in the previous point, involving certain lemmas (his/her mother—father) where appropriate.

Using these annotations the subjects can be categorized by the age they lived in, by profession, nationality or Location. All the RDF properties used in the process are in BioBase’s namespace, but they can be easily mapped into other namespaces, like FOAF if required.

4.2 FlatBase

IF recognition in FlatBase is also two-layered but in a completely different way than in BioBase. On the first level, the *main information* about the advertisement are attempted to be identified. Some examples:

- *Type of property*: House, Condominium, Apartment etc.
- *Location*: Part of the Country, City.
- Price range
- Size

When a piece of *main information* becomes available, a guided retrieval of details starts. Some examples:

- Details about the location (district or smaller area name, street name)
- Material used to build the property
- In case of flats : on which particular floor is the property; is this the top/ground floor
- Type of heating (a different list for different types of property)

On both levels specially configured JNER instances are used. The configuration initially includes a shorter list of locations, regular expression based rules to recognize price and size, contact information. IF recognition is then based on these known NEs and certain lemmas [2] including building materials, heating types, etc. When a piece of information becomes available, the JNERs may be reconfigured with respect to it (e.g. loading the area names when a Budapest district becomes known).

There is a list of required information configured in the system (basically the main information mentioned on above). When some information from this list are missing in the ad at the time of saving, the user is asked to write about them [3]. If this happens the second time, a limited number of direct questions are formulated about them, every question only once. After that the ad is saved even when there are still missing pieces of information.

5 Conclusions

5.1 Findings

Besides the many experiences earned in the area of NER, IFR and XML based software architecture, we have formulated some requirements against any annotation system based on suggestions. Yan Bodain and Jean-Marc Robert have composed five requirements on the static properties of semantic annotations [2]: **Robust anchors**, **Transparency**, **Liberty in choosing the semantic vocabulary**, **Variable Granularity**, **Handling dynamic updates**. We agree on that requirements[4] and the experiments with Docuphet allowed us to formulate some findings on the process of creating annotations by suggestion:

1. A particular question must never be asked twice as it is very frustrating for the user to answer the same question several times. This requirement has implications:

2. Every suggestion must be stored in the document, regardless the given answer. Per-session storing is not enough since the user may store the document and open it again later in the day. Further research is needed to find out whether this information should be stored on a per-document or a per-user basis.

3. This leads to two main types of suggestions: suggestions which turned into annotations on user confirmation and suggestions which aren't true (user returns negative answer). It would be an interesting research to find out how to make use of this "negated" information.

4. According to the rules described in [2], the annotations must be re-validated upon every text change. Given our point 1, this is a requirement very hard to meet. Theoretically it is possible to insert a statement into the first part of a

[2]The term "keyword" might be more appropriate in this case.
[3]But the current contents are stored nevertheless
[4]Though we use a pre-defined semantic vocabulary because of the nature of the system.

document which negates the meaning of everything or a particular part after or before it. To handle this appropriately, we should either fully understand the meaning of the change and update the right annotations or re-ask every question. The first solution is yet not possible to carry out, the latter causes a high number of undesirable questions. To get around this problem, we devised two techniques:

- **Limited scope**: We have defined two types of annotations: basic and derived. Basic annotations are only regarding to a portion of the text (a title, a paragraph, list item). We assume that changes elsewhere does not affect them. To ensure this as much as possible, basic annotations are very simple e.g. ”This paragraph is related to Budapest” when the token *Budapest* is present. This annotations are usually retrieved by NER as described in 4.1. These annotations are re-validated upon text change in the corresponding document part.
- **Dependencies**: There is a dependency graph of annotations. Derived annotations depend on basic or other derived annotations. The dependency is tracked with a list of annotation IDs. Every time an annotation changes, the dependent ones should be re-validated. If an annotation is deleted the derived annotations must be deleted as well. Furthermore, a derived annotation may require re-validation independently from the change of any other annotation.

5. The number of questions asked at once must be limited. This is important because if the user pastes in a larger portion of text tens of suggestions may be generated. These must be asked in more than one turn.

5.2 Comparison

There were different types of tools and applications examined before starting the Docuphet project

- **Semantic Wikis** One form of entering semantic annotations in documents is using semantic wikis. These applications enables the user to input RDF data by using a special syntax. Semantic Mediawiki, Artificial Memory [14], Kaukolu [4], PHPWiki [25], IkeWiki [19], SWiM [13] are semantic wikis. The available semantic vocabulary and the granularity of annotations vary in these applications but in all cases the wikis require semantic handling skills from the user.

- **Desktop ontology builders and annotators** There are some feature-rich desktop annotators for authoring semantically annotated documents. Protégé [17], TopBraid [24], Amaya [18], and Mangrove [15], are frameworks for building ontologies and knowledge graphs. SWEDT [16], Apolda [27], KATIA [2] have rich document editing and annotating capabilities. These are professional tools for knowledge experts.

> S-CREAM [10] integrates the Amilcare [3] IE module which implements semi-supervised machine learning: a set of training data must be annotated by hand in advance, then Amilcare is capable of creating certain annotations automatically in new documents.
>
> In COHSE [1] provides highlights and extra information for strings matched in a pre-defined knowledge base, Magpie [7] allows annotation of a pre-defined set of concepts based on forms.

Docuphet has many in common in the visualization of annotations with SWEDT or KATIA. Similarly to S-CREAM it uses IE technology. [5] Article storing is similar to some XML-based wikis. Like COHSE and Magpie, Docuphet is based on pre-defined concepts and relations which we call information frames. On the other hand, the fact that the target audience is the unskilled user, and the natural language suggestion element renders our solution rather unique.

5.3 Summary

Docuphet is a system for semantically annotating text with the contribution of the user. The process is facilitated by Information Extraction techniques: Named Entity Recognition and Information Frame Extraction (See Sections 3.4.1, 3.4.2).

In Docuphet — unlike some semantic wikis — it is not possible to annotate the text and build the corresponding ontology at the same time. Therefore Docuphet is only capable of handling pre-defined RDF triples, which is a quite inflexible property of the system. On the other hand, this very property allows to compose easy-to-understand questions about the known triples (as the questions are defined when an IF is defined). This way it is easy to create annotations even for the completely uninitiated users.

Given these properties Docuphet is most useful when the domain of the text is known in advance. Two exemplary applications are presented in Section 4. Other possible applications include annotation of economic or sport news, product reviews or Geolocation reviews. In this cases the pre-definition of IFs and rules to recognize them is required.

However, there is a basic functionality available without a specific knowledge about the domain. Docuphet is capable of recognizing named entities in an arbitrary text by using catalogs and formulating questions about the named entity candidates. This makes it very useful in building named entity databases and in disambiguation applications.

References

[1] S. Bechhofer, C. Goble, L. Carr, and S. Kampa. COHSE: Semantic web gives a better deal for the whole web? *ISWC International Semantic Web Conference Poster*, 2002.

[2] Y. Bodain and J.-M. Robert. Developing a robust authoring annotation system for the semantic web. *Proc. of 7th IEEE Int. Conf. on Advanced Learning Technologies*, 2007.

[5] but in a very different way

[3] F. Ciravegna, A. Dingli, Y. Wilks, and D. Petrelli. Amilcare: adaptive information extraction for document annotation. *SIGIR'02: Proc. of the 25th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 367–368, 2002.

[4] DFKI Knowledge Management. Kaukolu. `www.dfki.de/web/forschung/km/`.

[5] Docbook. `www.docbook.org/`.

[6] The Docuphet project. `www.docuphet.net`.

[7] J. Domingue, M. Dzbor, and E. Motta. Semantic layering with magpie. In *Handbook on Ontologies*, pages 533–554. Springer, 2004.

[8] FCKEditor. `www.fckeditor.net/`.

[9] GRDDL Working Group. Gleaning resource descriptions from dialects of languages. `www.w3.org/TR/grddl/`.

[10] S. Handschuh, S. Staab, and F. Ciravegna. S-cream-semi-automatic creation of metadata. *Proc. of the European Conf. on Knowledge Acquisition and Management*, 2002.

[11] HITEC. `categorizer.tmit.bme.hu/trac/wiki`.

[12] Á. Kenyeres. *Magyar Életrajzi Lexikon*. Arcanum Adatbázis Kft, 1994.

[13] KWARC. SWiM: A semantic wiki for mathematical knowledge management. `kwarc.info/projects/swim/`.

[14] L. Ludwig. Artificial memory. `www.artificialmemory.net/`.

[15] L. McDowell, O. Etzioni, S. D. Gribble, A. Y. Halevy, H. M.Levy, W. Pentney, D. Verma, and S. Vlasseva. Mangrove: Enticing ordinary people onto the semantic web via instant gratification. *Proc. of International Semantic Web Conference*, pages 754–770, 2003.

[16] R. G. Pereira and M. M. Freire. SWedt: A semantic web editor integrating ontologies and semantic annotations with resource description framework. *IEEE Int. Conf. on Internet and Web Applications and Services*, pages 200–200, 2006.

[17] Protégé. `protege.stanford.edu/`.

[18] V. Quint and I. Vatton. An introduction to Amaya. *Wide Web J.*, 1997.

[19] Salzburg Research. IkeWiki. `ikewiki.salzburgresearch.at/`.

[20] Gy. Szarvas and R. Farkas. Információkinyerés. In D. Tikk, editor, *Szövegbányászat*, chapter 4. TypoTEX, 2007.

[21] Szegedi Tudományegyetem Nyelvtechnológiai Csoport. Szeged korpusz 2. `www.inf.u-szeged.hu/projectdirs/hlt/`.

[22] D. Tikk. *Szövegbányászat*, chapter 2. TypoTEX, 2007.

[23] Tiny Moxiecode Content Editor (TinyMCE). `tinymce.moxiecode.com/`.

[24] TopQuadrant. Topbraid. `www.topquadrant.com/topbraid/composer/index.html`.

[25] VA Linux Systems. PhpWiki. `phpwiki.sourceforge.net/`.

[26] W3C Semantic Web Activity. Rfda. `www.w3.org/TR/xhtml-rdfa-primer/`.

[27] C. Wartena, R. Brussee, L. Gazendam, and W.-O. Huijsen. A practical tool for semantic annotation. *IEEE 18th Int. Conf. on Database and Expert Systems Applications (DEXA)*, 2007.