Docuphet – a dialogue-assisted content annotation tool

Mihály Héder
Budapest University of Technology and Economics
mihaly.heder@computer.org

Domonkos Tikk*
Institute for Computer Science
Humboldt University in Berlin
tikk@informatik.hu-berlin.de

ABSTRACT

In this decade the amount of textual content stored on the web became enormous, but the basic structure of web documents remained unchanged: a mixture of text and markup. When creating a document, the user rarely has the possibility of embedding semantic annotation into the content, because editor applications do not have such a feature or they are too difficult to use. We think that the creation of semantically rich documents can be best facilitated by a content editor with text mining technology running in the background. In this paper some of these technologies are brought into spotlight.

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 have never been able to produce the same quantity of content. The NuPedia project [24] that was started before WikiPedia is now accessible only in the Internet Archive. Citizendium [5], another controlled encyclopedia, has only some ten 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 the cases. Almost every traditional encyclopedia-like content repository uses some form of page formatting markup and plain text. Some of them offer a limited vocabulary to categorize the article, give the date, the creator, and specify some keywords.

There are research projects aiming to capture not only the formatting but also the semantics of the text at content editing. The majority of these applications define themselves as "semantic wikis". They enable the embedding of semantic annotation (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 language. This requires certain skills from the user and has a negative impact on the size of the potential audience.

Another approach is to annotate the text off-line, after the content has been created, without human supervision. To achieve this natural language processing (NLP) technology, namely information extraction (IE) tools are 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 average user to ease the creation of semantic annotations when editing web content. The Docuphet [9] project aims at discovering and experimenting with the possible solutions of the problem. In our view the web content creation should be a continuous, mutual dialogue between human and machine rather than a simple one-time input process.

This paper is organized as follows. In the following section we describe the motivations and goals behind Docuphet and the main components of the system. In Section 3 the technologies of the implemented components are detailed. Section 4 presents two example applications of the system. In Section 5 we discuss our experiences, and propose some requirements on dialogue-assisted semantic annotators. Section 6 reviews the related work in the field of semantic annotation software. Section 7 concludes the paper and discusses the future work.

2. THE DOCUPHET PROJECT

The main goal of the Docuphet project is to create a system that enables the user to produce semantic annotation easily. This is achieved via an intuitive user interface that is supported by text processing algorithms in the background. We intend to capture the meaning of the currently edited content by addressing simple questions to the user. While the user types, the available text is processed by IE applets in the background. Then extracted facts are formulated as statements that are conveyed to the user in the form of closed-ended questions. If the user confirms a statement, the system automatically embeds the appropriate semantic annotation into the text. The annotator software does not require any special technical knowledge or skill from the user, it is only assumed that she knows the particular content that she is editing, and hence she is able to decide if a related closed-ended question is true or not.

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 look-up operations are required to transfer the changes into the corresponding semantic annotations.

*On leave from Dept. of Telecommunications and Media Informatics, Budapest University of Technology and Economics

If the annotations were stored separately, the maintenance 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.[1] For the first applications, the Hungarian was selected as primary language.

Docuphet focuses only on the creation of annotations. The further use the semantically annotated content, such as semantic search and retrieval, or machine reasoning is currently out of the scope of the project. We rendered our efforts to the design and the validation of our concept, therefore some components of the prototypical applications are not yet optimized for the efficiency under heavy load.

2.1 The main components of Docuphet

The user interacts with the Docuphet Content Editor's (DCE) 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 suggests *annotation suggestion*s. Each suggestion comprises a textual statement or question, a numeric confidence level, one or more possible answers to the question, and semantic annotations (one per each positive answer). The confidence level is a real number of the unit interval, which specifies the validity of the suggestion. Under a certain (configurable) level suggestions are automatically disregarded. The module's suggestions are collected on the server and are sent back to the client, where they are presented to the user in the form of pop-up closed-ended questions. If the user confirms a statement the system inserts the corresponding 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 the components of Docuphet: the content editor, the annotation storage, the text processing and information extraction applets.

3. TECHNOLOGY OVERVIEW

3.1 The content editor

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 one for viewing the documents. At structured editors, the user handles objects like *section*, *title*, *paragraph*. This category includes wiki editors, publishing tools, DocBook 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 of the editor. There are two main categories: the more function-rich desktop applications that need to be installed on the client, and the web-based editors [11, 34] that require only a web browser to run. Web-based editors have been formerly simpler, but as AJAX become widespread, the complexity of such applications became almost equal to the desktop ones.

[1]In fact, this is one of the central problems of text processing to provide language-independent information extraction methods

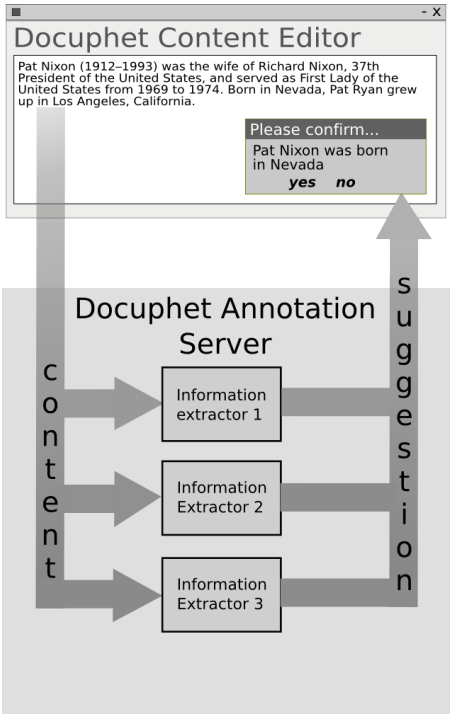

Figure 1: The overview of the components and the data flow of the Docuphet system.

The special requirements against DCE led to the development of a completely new solution. The DCE is an easy to use WYSiWYN[2] 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 the integration of annotations (see also Figures 4–5 for screenshots).

3.2 Content storage

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

1. simplicity to allow the implementation as web editor;
2. standard, stable and free to use;
3. proper support (documentation, examples, templates, editors, tools);
4. extendibility to carry annotations;
5. support of the following formatting: paragraphs, sections and titles; lists, program listings, emphasizes, images, tables, links.

Many options were evaluated: RTF , texinfo [13], troff [7], wikitext [42], XHTML, DocBook, DITA [25], LaTeX , ODF, and CDF [38].

Because of the large 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

[2]"What You See is What You Need" editors let the user to edit the structure of the document but not the source markup directly. They differ from WYSiWYG editors because further transformations are applied to generate the final view of the document.

in favour of DocBook, because this format is purely structural, it doesn't contain any markup relating to the document formatting, and it's grammar is defined in an easy-to-subtype Relax NG [26] format.

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 [39], GRDDL [14], Microformats and RDFa [40]. Finally we have selected RDFa because of its many advantages.

The RDFa [40] has been developed by W3C and by now it is a W3C recommendation. RDFa offers a technique that transforms 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 (an example is depicted in Figure 2):

```
<article
 xmlns="http://docbook.org/ns/docbook"
 xmlns:dc="http://purl.org/dc/elements/1.1/">
   <title property="dc:title">
      The trouble with Bob
   </title>
   <para id="ch1" property="dc:creator">
      Alice
   </para>
   <para about="#ch1" property="dc:creator">
      Tom
   </para>
   ...
</div>
```

Figure 2: Excerpt of an RDFa-annotated Docbook document

RDFa was also favoured because it is easy to integrate with other XML namespaces, 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 conform DocBook with RDFa, we extended the Docbook RNG schema to carry RDFa annotations, and we termed this DocBook profile DocBook/RDFa.

3.4 Extracting semantic information from the text

3.4.1 Named Entity Recognition

A named entity (NE) is a natural textual identifier of an object, such as e.g. person names, names of companies, locations, names of products, addresses, telephone numbers, email addresses. Named entity recognition (NER) is an NLP task, which aims at identifying and classifying NEs in the text.

The difficulty of the recognition task depends on the type of NEs. Telephone numbers, e-mail addresses can be easily recognized with simple regular expressions. The recognition of personal names and locations is more difficult, but it can be effectively supported with appropriate vocabularies. The recognition of company or product names can be much harder, because essentially no constraint applies on their surface form. NER is primarily performed by analysing some features of the candidate NEs, e.g. surface clues (capitalization, numbers, special symbols), the frequency, the in-sentence, in-paragraph, and in-document positions, whereas grammatical and morphological analysis may also be applied.

The Docuphet framework contains a general purpose named entity recognizer, the JNER, implemented in Java. This component comprises several modules, all of them analysing the text with a different technique. Many of them use vocabularies, e.g., for given names, company suffixes, locations. Others are based on regular expressions. It is possible to plug in external tools, such as a stemmer or a morphological parser. Other external tools can serve as connectors to databases (i.e IMDB, DMOZ) or to search engines (google, wikia search).

Although usually not considered as NEs, the recognition of professions (like painter, composer, engineer) and human properties (blond, tall) are also supported in JNER.

Conforming with the philosophy of Docuphet (ask relevant questions from the user), in this system the NER can also be assisted by asking closed-ended questions from the user.

3.4.2 Information Frame Recognition

Let us define an information frame (IF) as an RDF triple (we use Notation 3 syntax of RDF in this article for brevity)

```
<subject>
<predicate>
<object> .
```

in which at least one value is missing and thus substituted by a variable name. The class of the missing component(s) may be known. An example IF:

```
<X (an instance of the person class)>
<location of birth>
<Y (an instance of the location class)> .
```

The information frame recognition (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 above IF:

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

IFs can be defined in various ways. One method is to derive them from "semantic frames" which are used in the Berkeley FrameNet project [1] (see an example below). The aim of this project is to create an annotated lexical resource for English, by using frame semantics and supported by corpus evidence. These frames are referring to pre-defined conceptual structures.

```
frame(DESIGN),
inherit(CREATE),
frame_elements(DESIGNER(=CREATOR),
                       BUILDING(=WORK)),
scenes(DESIGNER desings BUILDING)
```

The frame elements are referring to certain semantic roles of actors and objects present in the scene. In the FrameNet project, human annotators mark the occurrences of the frame elements in texts.

Evidently, to find the possible semantic role of a text element, it is very helpful to have the named entities and their types identified

beforehand. In some cases additional rules may also be useful, i.e. specifying constraints on the relative position of the element of an IF (in-sentence, in-paragraph). While limiting the number of potentially identifiable IFs, this constraint has many practical advantages: it enables to start the IFR process before the whole content is available, and narrows significantly the search space thus reducing the computation time.

JFrame is the IFR component of the Docuphet framework, written in Java. IFs can be defined and the corresponding recognition rules can be given in JFrame. The input of the module is a token stream, in which the recognized NEs and their types are already marked. A JFrame IF definition may contain rules related to the class and lemmas of NEs in the token stream and have conditions on their order. JFrame also provides a confidence level for every recognized IF instance in accordance to the concept of Docuphet's workflow.

Currently in Docuphet, the IFR rules are defined by hand after experimentation. This is necessary because:

- There isn't enough properly annotated text which can be used as training data.
- The annotation method in Docuphet is based on dialogues. Therefore, for each IF a simple function must be defined, which generates the corresponding closed-ended questions.

3.4.3 Sentence segmentation

To support the recognition of IFs, a sentence segmenter was developed, named as JSentence. The component implements a modified version of the algorithm described in [33]. The Hungarian configuration of the tool was tested on the Szeged2 corpus [31], the biggest multi-thematic text corpus in Hungarian. The corpus contains 82096 sentences, and consists of complete novels from various authors and genre, high school essays, general newspaper articles, legal texts, computer related handbooks, and economic news. JSentence recognized sentence boundaries with a precision of 99.06 % at $FP \approx FN$. JSentence uses a rule-based algorithm with 15 regular expression based rules and 4 abbreviation lists.

4. APPLICATION EXAMPLES

In this section two demo applications of Docuphet are presented. BioBase is a web site to collect biographies of known people (similar to the ones in [18]), user autobiographies, or simple self-introductory texts. FlatBase is a real estate advertisement portal. In the case of the biographies, Docuphet is configured to recognize IFs based only on the entered text. For the FlatBase portal, the entered text is analysed first, and when some relevant information are still missing (e.g. floor number) Docuphet automatically asks questions from the user to complete the advertisement database properly. Both applications are configured to work on Hungarian text.

The workflow is the same in both scenarios as described in Section 2.1. Both applications uses DCE and the same document server component versions. They differ in the way how the annotations are produced, thus we discuss this part in detail next.

4.1 BioBase

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

1. JNER analyses 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 vocabulary, *location*, *nationality*, *profession*, *education*, *residence*, *family status*, *social relations (friends/other related people)* recognition based on vocabularies, *email* and *phone number* recognition based on regular expressions and *date* recognition based on a custom Java component, regular expressions and a list of month names.
2. JNER assigns a confidence level to every recognized NE, which is calculated by summing the pre-defined values of matching rules.[3] 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[4] that is set to 0.95. DCE accepts every annotation above 0.9 without asking a confirming question from the user, in order to avoid of flushing the user with questions.[5] 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 (see also Figure 4).
3. For the top three *person* and *location* NEs with confidence level between 0.8 and 0.95, an annotation suggestion of value

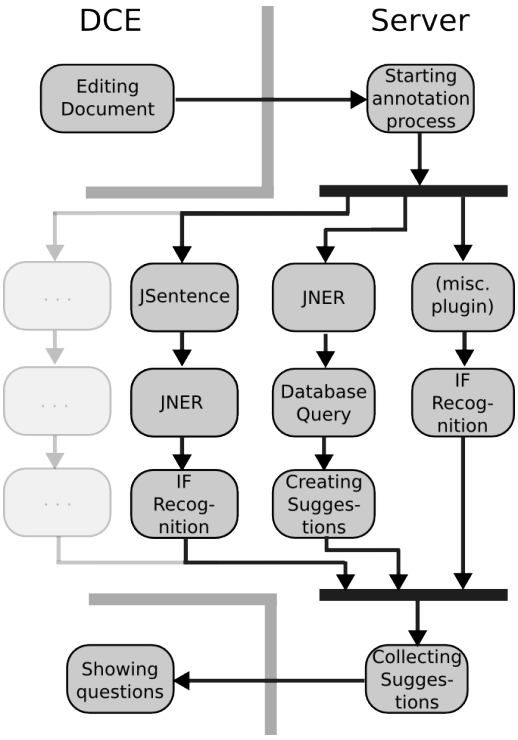

Figure 3: A typical workflow of Docuphet

[3]The creator of a rule can set both positive and negative confidence coefficients.

[4]This is based on the NE confidence level, but can be weighted, e.g. when a candidate person NE shares the surname with an already recognized NE.

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

Figure 4: A screenshot from BioBase

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 IE attempts are made:

1. First, the *person* is identified whom the article goes. This is done by creating suggestions on the first *person* NE found (in the title or in the text) with the triple
   `<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 mentioned *person*).
2. If the subject of the article is known, the date and place of birth and death are attempted to be identified next. This is done by analysing the *date* and *location* tokens in range of a centered window of 15 tokens around the occurrences the identified *person*. Extra confidence is added if the verb is in past tense form "született" (was born), "elhunyt | meghalt" (died) around a particular *date* or *location* (these are often omitted however).
3. *Nationality, profession, education, residence, friends* and *parent's name* are identified in a similar way as described in the previous point, involving certain terms (his mother | father) where appropriate.

Using these annotations the persons can be categorized by the era, which of they lived or live in, the profession, the nationality or location. All the RDF properties used in the process are in BioBase's namespace, but if required, they can be easily mapped into other namespaces, like FOAF [12].

Our experience with BioBase shows that Docuphet is able to capture the basic biographic information about a person. When creating a 500 word long biography (see Figure 4), the system pops up 10–15, mostly adequate questions. One direction of further work could be the to addition of new event IFs based on the review of biographies.

4.2 FlatBase

IFR performed in FlatBase also in two layers, but in a completely different way than in BioBase. On the first level, the *main information* of the advertisement are attempted to be identified:

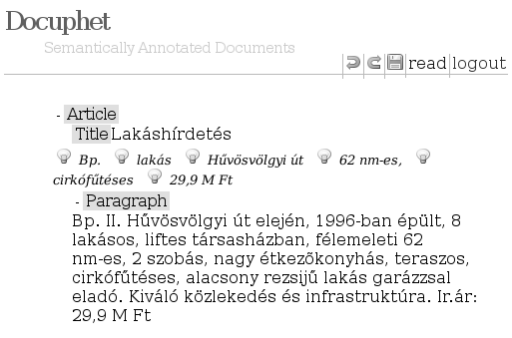

Figure 5: A screenshot from FlatBase.

- *type of property*: house, condominium, apartment etc.,
- *type of offer*: for rent, for sale,
- *location*: countryside, city,
- *price range*,
- *size*,
- *contact information*.

When a part of *main information* becomes available, the details are attempted to be extracted. Some examples:

- details about the location (district or quarter, street name);
- building materials;
- in case of flats: on which particular floor is the property; is this the top/ground floor;
- for certain types of property: orientation (street, garden);
- for a house: size of the garden;
- type of heating (depends on the property type);
- arrangement of rooms (e.g. separated entrance);
- public transport facilities (different types, based on the city).

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, size, and contact information. IFR is then performed based on already known NEs and certain trigger words related to building materials, heating types, etc. When some information is extracted, the JNERs may be reconfigured accordingly (e.g. loading the corresponding quarters when a district of Budapest is found).

When some of the main information are missing from the ad by saving, the user is asked to complete.[6] If this happens again, a limited number of direct questions are formulated related to the missing information, every question only once. At the end, the ad is saved even if some information are still missing.

FlatBase has proven to be very effective because flat advertisements are usually short, very similar to each other, and their vocabulary is limited. As a result, Flatbase is capable of extracting nearly all (usually not more than 5–10, see Figure 5) important information from an advertisement. This suggest that FlatBase may become a user-friendly input-assisting tool for advertisement portals.

5. DISCUSSION

[6]But the actual content is stored anyway.

Bodain and Robert composed five requirements on the static properties of semantic annotations [3]:

- robust anchors,
- transparency,
- freedom in choosing the semantic vocabulary,
- variable granularity,
- handling dynamic updates.

For dialogue-assisted annotators, such as Docuphet, most of the above requirements can be carried over[7], but as an outcome of our experimentation, we can now formulate additional requirements on the semantic annotation creation:

1. A particular question must never be asked twice to avoid of the discontent of the user. This implies that:
2. Every suggestion must be stored in the document, regardless from the answer, if any. Per-session storage is not sufficient since the document can be edited in several sessions. Further research is needed to find out whether suggestions should be stored on a per-document or a per-user basis.
3. There are two main types of suggestions depending on the positive/negative answer of the user. As a future work it should be investigated how negative answers can be exploited at annotation.
4. According to the "handling dynamic updates" rule described in [3], the annotations must be re-validated upon every text change. Given our point 1, this requirement can hardly be met. Theoretically it is possible to insert a statement into the first part of a document which negates the meaning of everything in a given scope. To handle this appropriately, we should either fully capture 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 regard only to a specific text scope (a title, a paragraph, list item). We assume that changes outside the scope do not affect them. To fulfil this assumption, basic annotations are very simple e.g. "This paragraph is related to Budapest" when the token *Budapest* is present. These annotations are usually retrieved by NER as described in Section 4.1, and re-validated upon text change in their scope.
   - **Dependencies**: We define 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, its dependants should be re-validated. If an annotation is deleted, the derived annotations must be deleted as well. Furthermore, a derived annotation may require a re-validation when non-annotated parts of the text change, since some derived annotations may depend on the characteristics of the text or on missing annotation. For example if the user accepts a new "main category" annotation, the old one must be re-validated or simply deleted.
5. The number of questions asked together has to be limited. This is important because if the user pastes in a larger piece of text, then many suggestions may be generated. These must be asked in several turns.

[7]although we applied a pre-defined semantic vocabulary because of the nature of the system

6. RELATED WORK

In the last 15 years, an excess of semantic annotation tools had been developed. Here we recall and compare the most important ones (see also Table 1).

We can divide the semantic annotators into two main groups:

- **Semantic Wikis:** One form of inserting semantic annotations into documents is via semantic wikis, such as Semantic Mediawiki [23], Artificial Memory [21], Kaukolu [8], PHPWiki [36], IkeWiki [30], and SWiM [19]. These applications enable the user to input RDF data by using a special syntax. The available semantic vocabulary and the granularity of the annotations vary in these applications, but in all cases semantic handling skill is required from the user.
- **Desktop ontology builders and annotators:** This group contains some feature-rich desktop annotators for authoring semantically annotated documents. Protégé [28], TopBraid [35], Amaya [29], and Mangrove [22] are frameworks for building ontologies and knowledge graphs. SWEDT [27], Apolda [41], and KATIA [3] have rich document editing and annotating capabilities. These are professional tools for knowledge experts.
  S-CREAM [15] integrates the Amilcare [4] IE module that implements a semi-supervised machine learning method: a set of training data must be annotated by hand in advance, then Amilcare creates certain annotations automatically in new documents.
  COHSE [2] highlights text and provide additional information for strings matching elements of a pre-defined knowledge base. Magpie [10] allows annotation of a pre-defined set of concepts based on forms.

Docuphet has many in common in the visualization of annotations with SWEDT [27] and KATIA [3]. But unlike these tools, Docuphet's editor hides the annotation markup details from the user and provides annotation visualization instead.

Like Melita [32], AKTiveDoc [20], MnM [37], or S-CREAM [15] Docuphet also uses IE technology to extract semantic annotation candidates from the text. However, the way Docuphet uses IE is quite dissimilar from these tools, since it uses IFs as a common concept of semantic information and involves the user into the process.

Like COHSE and Magpie, Docuphet is based on pre-defined concepts and relations that are termed information frames. The targeted audience of unskilled users and the NLP- and IE-based dialogue-assisted annotation creation render our solution rather unique. Our approach is comparable to the question-sequence-based guidance provided by some complex installation wizards, where the questions are based on the information gathered earlier. In Docuphet the set of IFR elements represents the knowledge base, which provides a sophisticated, flexible and order-independent solution.

7. CONCLUSION AND FUTURE WORK

Docuphet is dialogue-assisted semantic text annotator. The computer–human dialogue is facilitated by IE techniques: named entity recognition and information frame extraction. In Docuphet—unlike some semantic wikis—it is not possible to annotate the text

Table 1: Comparing Docuphet to other solutions

Name Semantic annotation input	Platform Semantic vocabulary	Editor type Way of storage	Storage format Skills required
WikiPedia –	Web –	textarea –	wikitext –
Semantic MediaWiki markup	Web arbitrary	textarea special wikitext	wikitext ontologies, markup
SWEDT form+RDF source	Eclipse arbitrary	source code editor RDF	HTML Web development, ontologies
Katia Drag-and-Drop	desktop Java predefined	word processor RDF server	HTML ontologies
Amaya forms	desktop application arbitrary	source code editor RDF server	HTML Web development, ontologies, RDF
Melita forms+named entity suggestions[a]	desktop application predefined	word processor RDF database	HTML ontologies
Docuphet suggestions in natural language	Web predefined	What You See is What You Need RDFa	Docbook –

[a]It collects all named entities as potential instances of ontology classes

and build the corresponding ontology simultaneously. Therefore Docuphet is only capable of handling pre-defined RDF information triples, which limits the flexibility 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 together with the corresponding IFs. 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 were presented in Section 4. Other possible applications include annotation of economic or sport news, product reviews or geolocation reviews. In these cases the set of appropriate IFs and corresponding IFR rules have to be created in advance.

Despite these limitations, there is a basic functionality available without specific domain knowledge. Docuphet is capable of recognizing NEs in an arbitrary text, and formulating questions about the NE candidates. This makes it a very useful tool for building NE databases and for disambiguation applications. Another possible application area is the assistance of context sensitive browser applications tools, such as In4's iGlue [17] and Context Discovery Inc.'s Context Organizer [6].

As for the future work, we intend to enable Docuphet to access and edit wikipedia articles via the interface provided by the MediaWiki's public API. As wikipedia uses the wikitext format, being very different from Docbook/RDFa, apparently the most problematic is the conversion of the articles, and the placement of the the annotations in wikitext. We also plan to integrate Docuphet with large public databases like IMDB, to facilitate disambiguation and named entity recognition.

We think that in machine understanding, bidirectional communication — questions and answers — is a key element — just like in human understanding. However, we admit that if the questions are not relevant enough, this proactive behavior probably causes discontent on the user's part. To find out more about the users' reactions when using our system, we intend to conduct experiments and surveys with many users.

The relevance of the questions can be improved if topical category labels are available for the documents. Therefore we plan to prepare the Docuphet to collaborate with document classifiers, such as e.g. the hitec3 framework [16].

Acknowledgement

Domonkos Tikk was supported by the Alexander von Humboldt Foundation.

8. REFERENCES

[1] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

[2] 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.

[3] 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.

[4] 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.

[5] Citizendium. `en.wikipedia.org/wiki/Citizendium`.

[6] Context Discovery Inc. Context Organizer for the Web. `http://www.contextdiscovery.com/context-organizer-for-the-web.aspx`.

[7] R. Corderoy. troff. `www.troff.org/`.

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

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

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

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

[12] The *Friend Of A Friend* project. `www.foaf-project.org/`.

[13] Free Software Foundation. texinfo. `www.gnu.org/software/texinfo/`.

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

[15] 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.

[16] HITEC. categorizer.tmit.bme.hu/trac/wiki.

[17] In4 Ltd. The iGlue project. http://iglue.com/beta/.

[18] Á. Kenyeres. *Magyar Életrajzi Lexikon ((Hungarian Biography Encyclopedia))*. Arcanum Adatbázis Kft, 1994.

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

[20] Vitaveska Lanfranchi, Fabio Ciravegna, Phil Moore, and Daniela Petrelli. Document editing and browsing in aktivedoc. In *DocEng '05: Proceedings of the 2005 ACM symposium on Document engineering*, pages 237–238, New York, NY, USA, 2005. ACM.

[21] L. Ludwig. Artificial memory. www.artificialmemory.net/.

[22] 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.

[23] Semantic Mediawiki. semantic-mediawiki.org/wiki/Semantic\%5FMediaWiki.

[24] Nupedia. *en.wikipedia.org/wiki/Nupedia*.

[25] OASIS DITA Technical Committee. Darwin information typing architecture. www.oasis-open.org/committees/dita.

[26] OASIS Relax-NG committee. Relax-ng. www.oasis-open.org/committees/relax-ng.

[27] 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.

[28] Protégé. protege.stanford.edu/.

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

[30] Salzburg Research. IkeWiki. ikewiki.salzburgresearch.at/.

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

[32] Advanced Knowledge Technologies. Melita. *howhttp://www.aktors.org/technologies/melita/*.

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

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

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

[36] VA Linux Systems. PhpWiki. phpwiki.sourceforge.net/.

[37] 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. In *EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, pages 379–391, London, UK, 2002. Springer-Verlag.

[38] W3C CDF Working Group. Compound document format. www.w3.org/2004/CDF/.

[39] W3C Semantic Web Activity. Rdf/xml syntax specification. www.w3.org/TR/rdf-syntax-grammar/.

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

[41] 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.

[42] wikitext. en.wikipedia.org/wiki/Wikitext.