An Approach to Automated On-line Annotation

Martin Adam

Institute of Informatics and Software Engineering,
Faculty of Informatics and Information Technologies,
Ilkovičova 3, Bratislava, Slovakia,
martin@atrip.sk

Abstract. In this paper we describe a solution for automated text annotation based on ontology. Proposed method consists of several steps which use different algorithms to recognize parts of text as instances of concepts from one or more ontologies. Described method can work with plain text as input, but in some cases it can have better performance with web pages. Found annotations can be used for further processing either for ontology enriching or determining location of annotations in text. This solution is a part of the NAZOU project, was verified within Pannda project and is further developed.

1 Introduction

Present web consists mainly of HTML documents which primary task is only to define a way of how are elements presented to the user. The task of recognition of the sense of shown data is mostly trivial for a person, but often very difficult, if not impossible, for a machine. That's why the goal of Semantic Web initiative is to put semantic to present web, so data on web can be accessed and understood not only by a human, but also by a machine. Annotation, especially automated, is crucial part and the main challenge of the Semantic Web effort.

Solution to automated annotation presented in this paper is developed within the NAZOU [1] and is used in Pannda [2] project. The main goal of Pannda project is to design a method to simplify navigation on a web page by enhancing the page content with useful annotations. This annotation process is based on known ontologies and user preferences. Annotations are added into the document on-line, whenever a page is accessed by web browser.

Considering this, proposed method was neither designed to find new concepts nor directly to expand any ontology (even when this is possible). It is merely finding (supposed) instances of known concepts to mark them for user.

Beyond this, we are also discussing here problem of ambiguities. After annotating a document, we have to find out, if there are some duplicates or overlapped annotations and decide which of them to merge and how to do it.

2 Text Annotation

The task of annotation is done within five steps by utilizing of four different algorithms. We can split these steps into two different groups. In the first one,

the aim is to find parts of processed text, which should represent (or match) concrete instances from given ontologies describing domain. The second group of our annotation steps aims to find parts of text which could be potentially instances of known concepts from given ontologies.

In the following sections we will discuss all steps in detail.

3 Recognition of Instances

This group of steps tries to find concrete concepts, which are already at least in one of give ontologies. The goal here is to find out if and where particular instances occur in processed text.

3.1 Recognition of Instances According to their Labels

Both processed text and labels of all instances from all ontologies are normalized. Under normalizing we understand splitting text (labels) into words, lowercasing them, transfering every word to its root form and removing all stop-words (e.g. a, an, of, the, it, and, on). After this is done, we create a list of so called key-words from instances. We use all words from all instances in all ontologies. Then we tag all words in processed text with information about all instances which contain this word. If all key-words from an instance are together (following each other in any order), this part of processed text is marked as found instance.

In truth, this is one of the easiest ways how to find a concrete instance. Nevertheless with enough instances in ontology it has very high successful rate and can provide helpful data to user. When we know exact instance from the ontology, we can assume there is more information about it in the ontology which can be presented to end user. For instance if we find in text instance *New York*, we assume that we can find also some more detailed data like population, location of this city etc.

3.2 Recognition of Instances According to Regular Expressions

Originally we used this method only for searching of concepts. But in some more or less rare cases it can be useful also for recognition of individuals. For example if we have in an ontology instance like *Full Time* when describing job offer properties, we can find it in text in more mutations. E.g., *40 hours a week* agreement (in Slovakia it equals a full time employment). Of course such problem could be solved also by adding several labels to the same instance. As said, this is for rare cases and we used it mainly to keep the annotation system as open as possible for future rules.

Disadvantage of this technique is that it needs to adapt the ontology. We need to specify what regular expressions should be used to find an instance. More detailed description of this method is in next chapter, as it is useful especially by concept-based search.

4 Recognition of Concepts

The second group of annotating steps aims to find concepts which are not directly in ontology. We use data available about concepts to predict if some part of the processed text could be instance of some of the concepts. As well as by instance recognition, the main goal (considering objectives of the original Pannda project, which main function is to mark annotated parts in text) is to find locations of supposed instances in text. Of course, we can use this approach also to fill new instances into ontologies.

4.1 Recognition of Concepts According to their Labels

This approach is based under several assumptions. We assume that the input text for annotation was a web page and that the text was processed by a parser which can give us data about html-classes names. (We used such parser in the Pannda project.)

A web page, if written correct, should be formatted with cascade styles. We will assume that we are processing such page. We also assume that creator of this page named all cascade styles reasonably (e.g. style which will format the appearance of publication author is really called *Author*).

When we have the necessary input from the parser, we proceed in a very similar way with the instance-label comparison. Names of HTML-classes as well as labels of concepts from ontologies are normalized. Then we compare these names with Cosine metric[1] to avoid mistakes by mistyping or different word order.

This method is working under several strong assumptions, so as expected, it does not find instances very often. But if it does, it is very reliable, as the HTML-class naming was done by author of the web page.

4.2 Recognition of Concepts According to Regular Expressions

This approach was used also in the OnTeA project [3]. Disadvantage of this technique is that it needs to adapt the ontology. We need to specify what regular expressions should be used to find an instance on a concept. We are defining so called search patterns, which are normal regular expressions. Even though in the Pannda project we worked only with English texts, these regular expressions can be marked with a language tag and for different languages different patterns will be used.

Suppose we have region ontology with the names of countries, cities, maybe streets, but also instances like *Location*. We also know that we are going to process pages containing job offers, which have common layout. In case we want to find location of the job, it is reasonable to assume that it will appear in text similar to this:

[1] http://www.dcs.shef.ac.uk/šam/stringmetrics.html#cosine

```
Company: Pineapple
Location: Bratislava
Phone: 02-12345678
Email: company@company.com
```

So if we have concept called *Location*, perhaps it would be reasonable to define regular expression "Location: ([a-zA-Z]+[ ]*[-A-Za-z0-9]*)" for it. We use this approach to find also other items in such text. Also more sophisticated expressions for finding email addresses and web pages links can be constructed.

4.3 Recognition of Concepts According to Language Patterns

This method is very similar to the previous one. However, there are some important differences. We are working again with regular expressions to find instances of concepts, but these expressions are not concept specific. They are language specific and we call them *language patterns*. Moreover, to create such patterns more easily, it is possible to use pseudo expressions, which determine which part of text should be considered to be the name of a concept, which is the name of an instance and which is neither of them.

Every language pattern must have two pseudo expressions. <INSTANCE> and <CONCEPT>. Here are some examples of language specific search patterns:

```
The <CONCEPT> <INSTANCE>
The <INSTANCE> <CONCEPT>
<CONCEPT> such as <INSTANCE>
such <CONCEPT> as <INSTANCE>
<CONCEPT>, especially <INSTANCE>
<CONCEPT>, including<INSTANCE>
<INSTANCE> and other <CONCEPT>
<INSTANCE> or other <CONCEPT>
<INSTANCE>, a <CONCEPT>
<INSTANCE>, an <CONCEPT>
<INSTANCE> is a <CONCEPT>
<INSTANCE> is an <CONCEPT>
```

We can also define the word class. E.g. <CONCEPT:*noun*> will match only when there is a noun at this position. Similarly, when we write <ANY:*verb*> it means, that there has to be a verb at this position and in this case it won't describe neither concept nor instance. To determine the word class we use QTag tagger [4].

Typical match for second pattern would be a sentence which contains text *The Carlton hotel* or *The Danube river*. These patters are regular expressions and all tags like <CONCEPT>, <INSTANCE> etc. are merely changed into corresponding regular expressions.

These rules are language specific, but in general they are valid for major part of concepts (they are more or less ontology independent). We can annotate text with these rules even without having to modify our original ontologies, as it is by some other methods described here.

5 Ambiguities elimination

Under the term ambiguity we understand state, when some part of text (typically one, two words) is marked multiple times with the same concept. This can occur very easily in our approach, as our method consists of several different searching methods, which can logically result into finding the same term several times.

We have to consider also partially ambiguities. It means that one part of text, as well as it sub-part, was marked as instance of the same concept. For example, in text "Danube river" one annotation method finds only word *Danube* as an instance of concept *River* and other method finds whole expression *Danube river* as an instance of the *River* concept.

Under different types of ambiguities belong also cases like – *Central Europe* vs. *Europe* (in text "Central Europe" we will find both) or *Slovak republic* vs. *Slovak* (language).

What we do not consider to be an ambiguity is when the word Danube is found as name of river and also name of a hotel in Slovakia. Of course (probably) only one of the meanings is correct, but the annotation itself is correct even if we find both. Deciding which sense is really related to actual text is beyond the research in our project.

In present state, we solve only the first type of ambiguities. In such case, all such annotations related to the same part of annotated text are merged into one annotation. Starting and ending location of annotation in text is chosen so that all original annotations are located within the merged one.

To solve the second type of ambiguities, we would have to define (or assume such definition exists) relations between concepts, which say which concept or instance is more concrete description of other. For example *Central Europe* is more specific than *Europe*, hence only *Central Europe* should be annotated.

6 Evaluation and Conclusions

To evaluate above described method, we designed a tool which annotates web pages loaded into a web browser. Browser loads the page and tells an annotation service (typically running on separate machine) to annotate the page. After the service finishes, browser shows annotations directly inside the original document. User can see all the parts of text which were marked as relevant to his subject of interest and he is also able to view more detailed information about most annotations. On the Fig. 1 three annotated places in document (marked with small icon inside text) and window which appeared after clicking at one of them are depicted. This window contains further information about annotated term. For more details you can visit the Pannda project web page[2].

In our evaluation we used region ontology describing countries, mountains, cities etc. We let the annotation service to annotate several (quite long) documents with themes from similar domain (e.g. History of Europe, History of USA).

[2] http://pannda.atrip.sk/

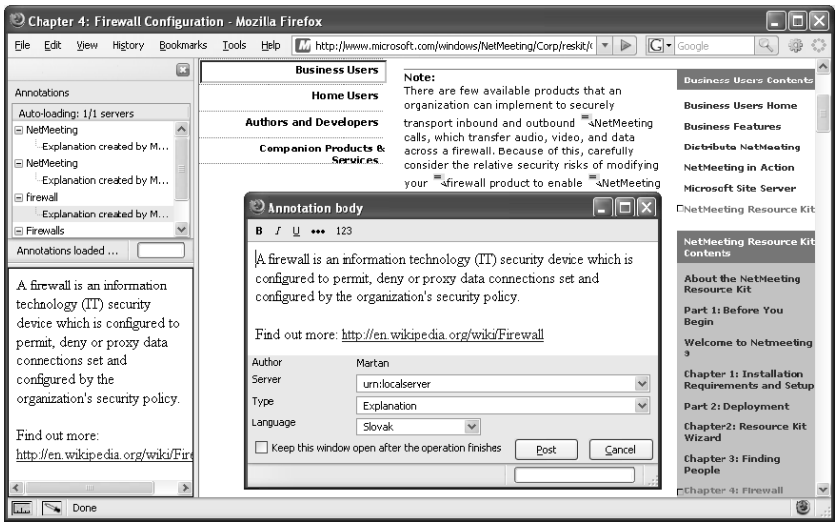

Fig. 1. Pannda annotations in FireFox

Then a person looked at the annotated page and marked wrong, redundant or missing annotations. Result of described testing showed about 80% precision rate by 70% recall rate.

The project is further developed and in the future work we would like to concentrate on fine tuning of the search patterns, especially language dependent ones. The ambiguities elimination will also be further explored and improved.

Acknowledgment

This work was partially supported by the State programme of research and development "Establishing of Information Society" under the contract No. 1025/04.

References

1. Návrat, P., et al.: Acquiring, Organising and Presenting Information and Knowledge from the Web. In Rachev, B., et al., eds.: Proc. of CompSysTech'06, Veliko Turnovo, Bulgaria (2006)
2. Adam, M.: Annotation of Accessed Web Pages During Accessing. Master's thesis, FIIT SUT in Bratislava (2007)
3. Laclavík, M., et al.: Ontea: Semi-automatic Ontology based Text Annotation Method. In Návrat, P., et al., eds.: Tools For Acquisition, Organisation and Presenting of Information and Knowledge, Bystrá dolina, Nízke Tatry, Slovakia, Vydavateľstvo STU, Bratislava (2006) 49–63
4. Tufis, D., M.O.: Tagging Romanian Texts: a Case Study for QTAG, a Language Independent Probabilistic Tagger. In: Proceedings of the First International Conference on Language Resources & Evaluation (LREC), Granada (Spain) (1998) 589–596