Apolda: A Practical Tool for Semantic Annotation

Christian Wartena, Rogier Brussee, Luit Gazendam and Willem-Olaf Huijsen
Telematica Instituut, P.O. Box 589, 7500 AN Enschede, The Netherlands
Christian.Wartena@telin.nl

Abstract

In this paper we give an overview of methods to find representations of ontology defined concepts in texts. We distinguish two approaches: lexicon-based methods and approaches using lexicalized ontologies. We focus on the latter method and describe the problems and choices that have to be made if this approach is put to work. Finally we describe an open-source tool that implements the lexicalized ontology approach along with two examples of its applicability in a practical context.

1. Introduction

Semantic annotation is a broad term covering several techniques to add labels to limited regions in a text referring to concepts with an externally defined semantics, usually from some ontology, taxonomy or thesaurus. The term semantic annotation thus includes topics like named entity recognition based on rules, heuristics or probabilistic models. The term also covers the relatively simple problem of finding literal occurrences of concepts in a text. This paper deals with the latter issue.

The problem of semantic annotation is tackled in a large number of projects. However, the problem is usually considered as part of the necessary evil that has to be done before the interesting problems can be addressed. As a consequence the solutions to this problem are seldom described in detail. For example Köhler et. al. [8] report on the detection of concepts in the ONDEX system, emphasizing semantic search (including disambiguation, heuristics and ontology mapping) while the look up problem, the first step of this process, is only sketched. In this paper we will give an overview of the problems of semantic annotation and the existing approaches to solve them. Moreover we describe a generic tool for semantic annotation that we have implemented which is already used in several projects.

The organization of this paper is as follows. In the first section we give an overview of the approaches to concept annotation that can be found in the literature. Here we identify two main streams. The first one enriches lexica with ontological information, the second one enriches ontologies with lexical information. In section 2.2 we describe the issues of the latter approach in more detail. In section 3 we present an open source tool that was recently released. Finally, in section 4 we describe two projects in which this tool was used.

2. Ontologies, Textual Representations and Semantic Annotation

2.1. State of the Art

For many tasks in text technology it is useful to relate parts of a text to concepts in an ontology. Such tasks include searching, keyword extraction, determining the subject of a text or comparing documents ([14]). Unfortunately, the relation between concepts and their possible textual representations is far from trivial. Almost everything can be a concept in an ontology, including very abstract concepts that might require long paraphrases, and the same concept may be described in several different ways. However, in most ontologies the majority of the concepts can be referenced in texts with only one word or a small fixed group of words. This might be related to the fact that most ontologies of practical importance are about rather concrete things. As a consequence it is often hard to distinguish between a word and the concept it represents. For example thesauri are often devised as a controlled keyword list, but they come with additional attributes such as synonymy, or hypernymy which indicate an ontological relation on the level of the implicit concept. Some terms are labeled as preferred terms indicating that it is the preferred textual representation of implicit context. A standard way to establish an explicit relation between concepts and their textual representations is to use a lexicon and an ontology, together with a relation between concepts of the ontology and the lemmas in the lexicon. This approach introduces an additional layer between the representation of a concept in a text and the concept in the ontology, namely the lemma. The advantage of this layer is that it allows to separate the problem of linking a lemma to a concept from the problems arising from the lemmatization of the text. In particular a concept itself can remain free of linguistic information such as part of speech and is language independent and concepts in the same or different ontologies can have the same textual representations. Furthermore we can link the same ontology to several dictionaries (e.g. for different languages). A good example of how lemmas and concepts can be related is constitutued by WordNet ([11]): each lemma belongs to a so called synset, a set of synonyms. Clearly, a word can have different senses. So a word can belong to a different synset. The synsets, rather than the words, are mutually related by the ontological relations of hypernymy, hyponymy, meronymy and holonymy and act as an explicit representation of the concept.

According to this approach, finding concepts in a text would involve two phases. First the text is lemmatized and subsequently the found lemmata are mapped onto concepts. This second step could either produce a list of all possible concepts (in the case of polysemy) or include a disambiguation procedure.

An approach in which information about the possible representations of a concept is included in the ontology is sometimes more practical than relating lemmas in a dictionary to an ontology. Creating and maintaining dictionaries and keeping them synchronized with an ontology, possibly under development, is not an easy task. In addition, in order to automatically find the occurrences of concepts in a text, the lexicon has to be usable by some lemmatizer which may impose additional constraints on the lexicon. These obstacles justify an approach in which information about the possible textual representations of a concept is included in the ontology itself. Ontologies (taxonomies) extended in this way are sometimes called *lexicalized* ontologies (taxonomies). They are used in influential projects like TAP ([6]) or KIM ([7]). Both projects use a knowledge base that is part of the ontology. The knowledge base is in fact a set of instances which have lexical variants, official name, etcetera as an attribute. In these lexicalized ontologies textual representations seem to always be linked to *individuals*, which has to do with the fact that the usage of the ontologies is strongly related to the recognition of named entities. In general however, we would also like to specify textual representations of classes like in, e.g., the model of [13].

The current approach is in line with the guidelines for ontology development given by the SKOS standard ([10]). In the SKOS RDF binding there are the distinct notions of a `skos:Concept` and two attributes to express the textual representation of a concept, the `skos:prefLabel` property for the preferred or standard representation and the `skos:altLabel` property for alternative representations. Both label properties are subproperties of the `rdfs:label` property. The recommended practice of the use of `skos:Concept` is that each term in a thesaurus that occurs as recommended label corresponds to a `skos:Concept`. In this model, relations defined on the semantic level like broader and narrower are defined on concepts while two labels of the same concept are (by definition) synonymous.

The ISO topic maps standard ([5], [12]) is designed for creating thesauri and indices of documents. Therefore topic maps make a strict distinction between a topic and the occurrence of a topic in a document. A topic can be anything including a concept. Topics can have different base names, sort and display names corresponding to different textual representations. When a topic “occurs” in a document is a decision that is left to the creators of the index, but clearly one strategy is to record the occurence at a specific location in a text of a basename of a topic.

2.2. Semantic Annotation with Lexicalized Ontologies

2.2.1 Level of abstraction

A text can be looked at at different levels of abstraction. Depending on our point of view, we can consider it as a sequence of letters, as a sequence of tokens or as a sequence of lemmas. This makes a difference, since at each level the text is normalized in some way, and ambiguities are resolved or introduced. It is not clear *a priori* at which level the textual representations of concepts should match. At a high level we can abstract from many problems related to capitalization, line breaks, hyphenation etc. On the other hand we also lose information or make it less accessible. This can be illustrated with e.g. the difference between a hyphen and a dash. A dash is normally separated by blanks from the preceding and following word. A hyphen on the other hand, is a separation marker by itself and has no surrounding blanks. If we treat a sentence as a list of tokens, this difference is no longer directly accessible. At this level of abstraction the string “Art - Works”, like in the web site title “The Metropolitan Museum of Art - Works of Art”, could readily become indistinguishable from “Art-Works”.

A similar problem occurs with capitalization. Usually capitalization is irrelevant and life becomes easier if we can abstract it away. However, there exist cases where the use of capitals makes an essential difference.

Likewise for lemmatization, if we apply matching at the level of (normalized) lemmas we can reduce the number of textual representations needed. For example, there is no need to specify the plural and the singular form of a noun and in languages with rich morphology we can reduce the number of representations even further. Unfortunately, case or number can make a difference as in “British Museum” vs. “British museums”. Again this difference could be made accessible as an attribute at the cost of complexity in the representation of a concept.

A way out of these problems is to make details which are abstracted from accessible as features of the higher-level elements. For example, the type assigned to a token by the tokenizer (e.g. dash or hyphen) can be coded as an attribute of the token. To use this extra information would require that the textual representation of a concept is no longer a simple string but an expression that specifies attributes of tokens. Unfortunately this could be a hindrance for using ontologies in which some kind of label or lexical representation is already present.

2.2.2 Other typical problems

Most of the problems mentioned above are related to handling so called *multiword expressions*. In lexicon-based approaches these are often somewhat problematic. Reasoning from the lexicon there is seldom a natural reason to code multiwords. For example, there is no need to code an entry like "art museum" for a task like lemmatization since both words are coded separately. In fact, such tasks will even be complicated by the addition of multiwords, because ambiguity is created. Moreover, in this way we either introduce redundancy (in this case of morpho-syntactic features of "museum" and "art museum") or dependencies in the lexicon (between "art museum" and "museum"). In the approach using lexicalized ontologies the problems of stemming/lemmatizing and the recognition of multiword entries is separated in a very natural way.

In either approach concepts can only be coded and recognized if they can be represented by a fixed phrase, typically a noun phrase. One could argue that a sentence like "He was soon producing some spectacular and original images" refers to the concept of "painting", but finding such references is far beyond the scope of both approaches.

Sometimes there is *ambiguity*, in the sense that two or more concepts have a common textual representation or the representations of two concepts in a text have an overlap. Usually, in these cases only one of the possible annotations is the contextually appropriate one. In our approach we separate the tasks of finding representations of concepts and disambiguation into two steps. The tool described below only does the first task. A next stage of analysis would have to solve the ambiguities (see e.g. [9]).

Even though a lexicalized ontology contains the information of the *proper names* which you want to recognize, this recognition is not simple. Often the ontology only contains the most complete representation (e.g. William Jefferson Clinton), whereas the number of ways in which reference can be made to this person is numerous: e.g. "William J. Clinton", "Bill Clinton", "governor Clinton", "president Bill Clinton", "president Clinton" and "Mr. Clinton." One approach is to store all properties of the person names of your ontology in lists: defining the first names, the last names and roles and to use grammatical rules for spotting combinations of elements. The applicable rules can annotate the found name with a normalized form which subsequently is matched with the ontology. This method allows for the recognition of many names, but the mapping still requires some effort. Another more general method is to generate all the variants of one name and store them in our lexicalized ontology. This name generation can be achieved by using grammatical rules. The advantage of this approach is that it allows for a direct annotation without any mapping. However, it does not allow for the recognition of new names or the slightest deviations of stored names.

3. Implementation

3.1. GATE

GATE is an open source framework for annotating texts developed by the natural-language processing group of the University of Sheffield ([4]). GATE manages annotations, language resources (like documents and ontologies) and processing resources (PR). Processing resources can access the annotations added by processing resources run previously and add new ones. The annotations are stored and managed by GATE. PRs can be build as Java plugins using the the APIs provided by the GATE framework. A number of common processing resources, like a tokenizer and a sentence splitter are part of the distribution. Two special PRs are the JAPE transducer and the gazetteer. The gazetteer takes lists of strings as input. Each list has a label. For each occurence of a string from a list, this occurence will be annotated with the label of the list. The JAPE transducer takes a grammar based on regular expressions as its input. The grammar specifies patterns over strings and annotations, and the annotations that should be added in case the pattern matches. Both PRs can be made aware of an ontology that is loaded as a language resource ([2], [3]). Each gazetteer list can be associated with a concept from the ontology. The patterns of the JAPE grammar can contain concepts of the ontology as well. The latter has to be interpreted in the sense that a pattern matches if any subconcept of the specified concept matches.

The usage of gazetteer lists with associated ontological concepts reflects the idea of a lexicon with a mapping of lemmas onto concepts as described above. The method is well suited for the treatment of large lists like lists of city names, and the like. For the recognition of a large number of different concepts with only a few representation per concept this method is not very feasible.

In the next section we will describe a PR that is able to annotate a text with concepts of an ontology by using specified labels for textual representation from that ontology.

3.2. Apolda

We have implemented our ideas described above about the way annotation with lexicalized ontologies should be done in a freely available plugin for GATE, called *Apolda* (Automated Processing of Ontologies with Lexical Denotations for Annotation). Apolda is published under the GNU Lesser General Public License (LGPL) at `http://apolda.sourceforge.net`.

Apolda can annotate texts on the basis of an OWL ontology that is loaded as a resource in GATE. The textual representations in the ontology should be coded as OWL annotation properties (`owl:AnnotationProperty`). Which annotation properties should be used for annotating the documents can be specified by the initialisation parameters. One can either specify a predefined property, such as `rdf:label` or `rdf:comment` or a user defined one. Two annotation properties can be specified, one for the prefered representation and one for the alternative representations. In the examples below we will discuss how to deal with ontologies using more than two different labels for the specification of textual representations. If no annotation property is specified in the initialisation of the plugin the identifier of the classes (`rdf:ID`, excluding namespace) is used as a substitute.

As described above, we have to make a choice for the level of abstraction at which an annotation tool works. In order to offer maximal flexibility while at the same time preserving simplicity in expressing textual representations Apolda supports two types of textual representations. The literal representations that have to match literaly at the lowest level of representation and the standard representations that can match at any level available. Moreover, standard representations are matched case insensitively while the literal ones are compared case sensitively with the document. In both cases only whole words will match, i.e. the matching region has to start at the beginning of a token and has to end at the end of a token, according to the tokeniser used. A literal textual representation matches a region in the text if it matches either the literal string of a token or, if a lemmatizer was run before, the lemma for that token. The literal representations can be specified in an intuitive way by using quotation marks. If a standard textual representation consists of more than one token each token can match at a different level. E.g. a representation *exhibiting painter* will match the string *exhibiting painters* that consists of tokens with lemmas *exhibit* and *painter*. Thus the first word will match at the string level, while the second one matches at the lemmatized level. If only a subset of the possible realisations should match, an exhaustive list of literal representations has to be specified.

Apolda will find all matches that are possible without trying to disambiguate. In our approach disambiguation should be delayed to a separate module. Thus Apolda can give several annotations to the same region or two annotations can have a partial overlap. Only one exception is made to this principle. If an annotation could be added to a proper substring of another annotation, only the longest match is added. Thus if *Rembrandt* and *Rembrandt van Rijn* are both representations of the same concept, the concept art:RembrandtVanRijn is added only once if the longer representation is found in a text.

Sometimes it is usefull to have textual representations for several languages in one ontology. In this case it is important to keep the representations from the different languages apart: the same string can denote completely different concepts in different languages. In OWL this is solved elegantly by allowing a language attribute on an annotation property. Apolda uses these language attributes, and will only use those textual representation for which the language attribute is consistent with the globaly specified language.

4. Practical Applications

We have used Apolda in an experiment to automatically generate keywords of TV programs in the archive of Sound & Vision, the Dutch national TV and radio archive. In this task we tried to generate the keywords for the TV program by analyzing its web site's text. In this archive, programs are currently described manually. The description consists of multiple sorts of information, of which one is the subject of the program described in keywords. The keywords are selected from a thesaurus containing 3700 terms. A SKOS representation of this thesaurus is available in RDF OWL. For one term multiple textual representations are specified: the plural form of the preferred spelling is in the `skos:prefLabel`, the singular form of the preferred spelling is in the `altLabelSg` and synonyms are in the `altLabelSyn` (plural) and altLabelSynSg (singular form). All altLabels are `rdfs:subProperty` of the `skos:altLabel`. By specifying the `skos:altLabel` and `skos:prefLabel` in Apolda, we will find all defined textual representations of the concepts. The thesaurus contained 9168 textual representations in total. Using Apolda we were able to annotate five hundred texts with all occurring terms in half a day. This half day was needed to make some small corrections in the OWL file, load all text, Apolda and the OWL file in GATE, specify the GATE pipeline and process the documents. We manually compared results against an existing method to spot terms in which we used GATE with gazetteer lists. Small differences where found in performance. A term which was not spotted by Apolda turned out to be a modeling error in the thesaurus which was incorporated into the gazetteer list but which was not remodeled into a term in the OWL file. The direct mapping of Apolda to the proper terms makes it clear that some words refer to multiple terms. This is not clear in the old process The biggest difference is the ease of use of Apolda. Apolda spots and annotates a term with the preferred terms URI without the need for any programming.

A second use of Apolda is in the MultimediaN ViTa project at the Telematica Instituut dealing with the semantic annotation of video material. One of the approaches used is automatic enrichment of text with metadata that is either already associated with the content or user generated. In this particular application, enrichment is done by automatically hyperlinking occurrences of Wikipedia concepts in the metadata text. For this application, the Dutch Wikipedia was used. To each Wikipedia article one or more "categories" are associated. E.g., the article on "Alan Turing" is associated with, both "Brits informaticus" (British computer scientist) and "Geschiedenis van de computer" (history of computing). This relation forms a hierarchical, ontology-like structure. Subsequently, this ontology, transformed into OWL, was read into Apolda, that could then annotate occurrences of the concepts in the metadata texts.

5. Conclusions and Related Work

The main contribution of this paper is its focus on and explicit elaboration of finding occurrences of concepts from an ontology in a text. Usually, this is part of a larger project and is paid little attention to as an independent problem. Thus, many design choices are not made explicit and the method is not available as a generic reusable component. In this paper we have described the problems of annotating texts with concepts from a lexicalized ontology. We have presented a publicly available tool that was developed on the basis of clearly defined principles and have shown its practical value in two concrete examples. Since the tool we have described here implements one of the two standard ways to link texts to ontologies, we believe that this tool is a useful extension of the GATE framework.

The two representation types, standard and literal, result in a great flexibility and offer sufficient possibilities to handle most lexicalized ontologies. In some cases, however, we would like to have even more expressivity to code the representation of more complex concepts (see e.g. [14]). Thus we consider implementing JAPE expressions as a third type of representation.

Finally, we want to mention tOKo, a tool that has large similarities with Apolda ([1]). tOKo aims to detect concepts in texts and offers a huge number of features to facilitate this. Semantic annotation with a lexicalized ontology is one of them. In tOKo there is no clear distinction between the concepts and their textual representation, but there are convenient ways to specify abbreviations, miss-spellings, synonyms etc. It is also possible to use patterns and link them to concepts. Internally tOKo works different from Apolda, since it does not try to match the representations at a lemmatized representation of the text but internally expands each representation to all possible surface forms.

Acknowledgements

This work was partly financed by MultimediaN and Catch. We thank William van Dieten for helpful comments.

References

[1] Anjo Anjewierden et al. tOKo and Sigmund: text analysis support for ontology development and social research. http://www.toko-sigmund.org, 2006.

[2] K. Bontcheva. Open-source Tools for Creation, Maintenance, and Storage of Lexical Resources for Language Generation from Ontologies. In *Proceedings of 4th Language Resources and Evaluation Conference (LREC'04)*, 2004.

[3] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. Evolving gate to meet new challenges in language engineering. *Nat. Lang. Eng.*, 10(3-4):349–373, 2004.

[4] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[5] L. M. Garshol and G. Moore. ISO 13250-2: Topic Maps — Data Model. Final draft, ISO/IEC, 16. December 2005.

[6] R. Guha and R. McCool. TAP:a Semantic Web platform. *Computer Networks*, 42(5):557–577, 2003.

[7] A. Kiryakov, B. Popov, D. Ognyanoff, D. Manov, A. Kirilov, and M. Goranov. Semantic annotation, indexing, and retrieval. In D. Fensel, K. P. Sycara, and J. Mylopoulos, editors, *Int. Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 484–499. Springer, 2003.

[8] J. Köhler, S. Philippi, M. Specht, and A. Rüegg. Ontology based text indexing and querying for the semantic web. *Knowl.-Based Syst.*, 19(8):744–754, 2006.

[9] V. Malaisé, L. Gazendam, and H. Brugman. Disambiguating automatic semantic annotations with thesaurus structure. accepted at TALN2007, 2007.

[10] A. Miles, B. Matthews, D. Beckett, D. Brickley, M. Wilson, and N. Rogers. Skos: A language to describe simple knowledge structures for the web. In *XTech 2005 Conference Proceedings*, 2005.

[11] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4):235 – 244, 1990.

[12] S. Pepper. The tao of topic maps - finding the way in the age of infoglut, 2000.

[13] D. Vallet, M. Fernández, and P. Castells. An ontology-based information retrieval model. In A. Gómez-Pérez and J. Euzenat, editors, *ESWC*, volume 3532 of *Lecture Notes in Computer Science*, pages 455–470. Springer, 2005.

[14] C. Wartena, A. Anjewierden, and W. van Dieten. Using patterns of higher order concepts to detect inconsistencies in large text collections. In *Proceedings of the International Workshop: Ontologies in Text Technology (OTT'06)*, pages 107—112, Osnabrück, 2006.