NLP-based support for ontology lifecycle development

Diana Maynard and Adam Funk and Wim Peters

Department of Computer Science
University of Sheffield
Regent Court, 211 Portobello
S1 4DP, Sheffield, UK

Abstract. This paper describes the implementation of an approach to modelling the dynamics of the propagation of textually derived semantic information, in particular with respect to networked ontologies. On the one hand, new ontologies may be generated automatically from textual data, or existing ontologies may be modified or extended according to new evidence. This can cause problems for large or networked ontologies, where only a small section of the ontology may be modified, and where multiple users may be working with the ontologies simultaneously. Collaborative editing of both documents and ontologies is becoming more widespread, but can bring major problems with change management: both within the ontology itself and between text and ontology. In this paper, we describe a plugin for the NeOn toolkit which uses both automatic and manual methods for generating and modifying ontologies on the fly from unstructured text, and enables two-way ontology lifecycle development.

1 Introduction

Ontology evolution is increasingly acquiring research momentum in the Semantic Web field. This is due to the fact that ontologies, forming the backbone of Semantic Web systems, need to be kept up-to-date so that ontology-based systems remain usable. Furthermore, collaborative creation and editing of documents such as Wikipedia and corporate or group wikis is becoming more and more widespread; however, it is difficult to extract the relevant keywords and facts from such documents when they are constantly changing and when a single user is not responsible for their maintenance. Similarly when ontologies change, this may affect systems dependent on them. In a collaborative ontology development environment, changes tend to be more frequent.

In this work, we describe an approach to modelling the dynamics of ontology and metadata change in text, motivated by the need for constant ontology evolution and for the propagation of changes in an ontology to its instances and properties. For example, if a user deletes concepts from the ontology, it is important to have a mechanism for dealing with any associated semantic metadata,

in order not to lose vital information. If a user adds new concepts to the ontology, then it may be necessary to return to the text to check whether additional instances can be found which should be used to populate these new concepts in the ontology. We call this the top-down approach to ontology change.

On the other hand, not only are ontologies dynamic and subject to structural change, but so are the texts and instances from which the ontologies may be derived. If we get additional relevant textual material and/or find new instances in that text, it may be necessary to modify the ontology to take into consideration this new information (for example, adding new concepts or new relations between existing concepts in the ontology). We call this the bottom-up approach to ontology change.

We have developed a generic GATE application[1] called SPRAT (Semantic Pattern Recognition and Annotation Tool) which uses ontology-based information extraction techniques [1, 2] to generate ontological information from unstructured text and either create a new ontology from scratch or augment an existing ontology with new entities. This application represents a typical situation where NLP (natural language processing) techniques can assist in the development of Semantic Web technology. There are many NLP tools used in both research and industry, of which GATE is one of the most widely used and well known. There are also a variety of tools and systems providing ontology lifecycle support, of which the NeOn toolkit is one newly emerging example. However, one of the current research challenges is to provide better interoperability between these two rather different kinds of systems. For example, GATE provides ontology support but it is limited in comparison with with ontology editors and frameworks such as Protégé [3] and the NeOn toolkit[2]. SPRAT is available as part of the GATE webservices plugin for the NeOn Toolkit[3].

Both the top-down and bottom-up approaches combine support for ontology lifecyle development and NLP engineers. The ontology refinement and change log mechanisms described in this paper enable ontological changes to be made in GATE by language engineers and domain experts who are interested in acquiring new information to populate an ontology, but who do not necessarily know or care how this information may later be repurposed. These people will use GATE to analyse the documents and/or create the data, but are not interested in ontology management per se. On the other hand, ontologists who are not concerned with where the data comes from, but only what use they will make of it, do not have to worry about how to get hold of the relevant information in the first place or details about language engineering tools.

2 Related work

In this work, we use lexico-syntactic pattern-based methods for ontology population. The idea of acquiring semantic information from texts dates back to the

[1] General Architecture for Text Engineering, `http://gate.ac.uk/`
[2] `http://www.neon-toolkit.org`
[3] `http://gate.ac.uk/projects/neon/webservices-plugin.html`

early 1960s with Harris' *distributional hypothesis* [4] and Hirschman and Sager's work in the 1970s [5], which focused on determining sets of sublanguage-specific word classes using syntactic patterns from domain-specific corpora. A detailed description and comparison of lexical and syntactic pattern matching can be found in [6], In particular, research in this area has been used in specific domains such as medicine, where a relatively small number of syntactic structures is often found, for example in patient reports. Here the structures are also quite simple, with short and relatively unambiguous sentences typically found: this makes syntactic pattern matching much easier.

Using lexico-syntactic patterns (LSPs) has previously proved to be successful for a variety of tasks[7]. Various attempts have been made to refine the patterns in a semi-automatic way, for example using the web as evidence [8]. Other methods focus mainly on a specific kind of pattern, such as *part-of* relations [9], or use clustering approaches [10]. The disadvantage of the latter is that they require large corpora to work well and generally fail to produce good clusters from fewer than 100 million words.

The closest approach to ours is probably Text2Onto [11], which performs hyponym extraction on the basis of patterns. It combines machine learning approaches with basic linguistic processing such as tokenisation or lemmatisation and shallow parsing. SPRAT differs in that it has a greater number of LSPs, and it currently uses only a rule-based approach rather than machine learning, with no statistical clustering or parsing. This leads to much increased precision over Text2Onto, though fewer relations are produced. It also enables a more flexible approach and fine-tuning of the system.

We also took inspiration from some currently unpublished research carried out at DFKI in the Musing project[4], which aims to derive T-Box Relations from unstructured texts in German. In this work, attention is focused primarily on deriving relations between parts of German compound nouns, but we can make use of similar restrictions.

Within the range of activities required for ontology learning, SPRAT covers a number of intermediate stages in the process of ontology acquisition, namely term recognition and relation extraction. In the initial acquisition stage, it will recognise terms from the corpus only if they participate in any of the patterns. This guarantees termhood only up to a certain extent. For relation extraction, SPRAT does not make use of a parser. There are many applications that make use of syntactic dependencies, e.g., [12, 13]. Our approach differs from this in that our patterns are defined at low levels of syntactic constituency, such as noun phrases, and by means of finite state transducers. Identifying and engineering on the basis of the linguistic building blocks that are relevant for each ontology editing task eliminates the need for a parser. This bottom-up approach is much faster and less error-prone than a parser, and is more in line with the ontology bootstrapping approach advocated in [14].

[4] `http://www.musing.eu/`

3 Ontology acquisition using SPRAT

In this section, we describe the bottom-up approach to ontology change, using the SPRAT application. We combine aspects from traditional named entity recognition, ontology-based information extraction and relation extraction, in order to identify patterns for the extraction of a variety of entity types and relations between them, and to re-engineer them into ontological concepts, instances and properties. The application is developed in GATE. Basic linguistic patterns are first identified, and then enhanced with deeper semantic knowledge. The NEBOnE plugin for GATE is used in order to generate the new ontological entities on the fly: this is described in more detail in Section 3.3.

SPRAT makes use of a number of pre-existing GATE resources for preprocessing the text with linguistic information, such as tokenisation, part-of-speech tagging, simple noun phrase and verb phrase chunking, and named entity recognition. It also includes lexical resources containing semantic classes from WordNet [15] and VerbNet [16], which enable the incorporation of deeper semantic information. This allows us (i) to look for verbal patterns connecting terms in a sentence, using the ANNIC plugin in GATE [17], and (ii) to restrict the kinds of relation extracted. For example, we can restrict the kinds of entities that have body parts associated with them to animals and humans.

3.1 Identifying Linguistic Patterns

Identifying ontological concepts and/or relations in text requires a slightly different strategy from that used in traditional named entity recognition and ontology-based information extraction methods [1]. While we can still make use of known terms (either via a gazetteer or by accessing the class, instance and property labels in an existing ontology), this is often not sufficient because the concept might not be in the ontology, or only as a synonym or variant. We therefore make more use of linguistic patterns and also contextual clues, rather than relying on gazetteer lists. We have identified three sets of patterns which can help us identify concepts, instances and properties to create a new ontology or extend an existing one: the well-known Hearst patterns [18], the LSPs developed in the NeOn project corresponding to Ontology Design Patterns [19], and some new contextual patterns defined by us [20]. The ontology building process is dynamic, so we can make use of the initial concepts and instances generated in latter stages of the process in order to further refine the ontology. We have extended the patterns described in [20] to incorporate further semantic restrictions and to include more properties. An example of a linguistic pattern is the following:

Add a new class. Here we postulate that an unknown entity amidst a list of known entities is likely to be also an entity of the same type. For example, if we have a list of classes of fish, and there is an unknown noun phrase in amongst the list, we can presume that this is also a class of fish. To decide where to add this new class in the ontology, we can look for the Most Specific Common Abstraction (MSCA) of all the other items in the list (i.e. the lowest common

superclass of all the classes in the list) and add the new entity as a subclass of this class.
Example: *Hornsharks*, *leopard sharks* and *catsharks* can survive in aquarium conditions for up to a year or more.
where *hornshark* and *leopard shark* are classes in the ontology and *catshark* is unknown, so we can recognise *catshark* as a subclass with the same parent as that of *hornshark* and *leopard shark*, in this case *shark*.

An example of selectional restrictions placed on a relation is that a property must be restricted to the following semantic type from WordNet: plant, shape, food, substance, object, body, animal, possession, phenomenon, artifact. Experiments with this restriction improved precision from around 35% to 75%, although unsurprisingly, recall dropped a little.

3.2 Implementation of patterns

All three sets of patterns are implemented in GATE as JAPE rules [21]. On the left hand side of the rule is the pattern to be annotated. This consists of a number of pre-existing annotations which have been created as a result of preprocessing components (such as POS tagging, gazetteer lookup and so on) and earlier JAPE rules. The right hand side of the rule invokes NEBOnE and creates the new items in the ontology, as well as adding annotations to the document itself.

The following snippet shows the left hand side of a JAPE rule to match the *sharks* example described above:

```
Rule: FishList1
({FishClass}(AND)  ({NP}|(LIST)) ):mention
--> ...
```

This pattern looks for something that has been identified in the fish ontology as a class, followed by either a single noun phrase or a list of noun phrases separated by conjunctions or commas. On the right hand side of the rule, NEBOnE is invoked and each Noun Phrase not already listed in the ontology is added as a new class, at the same level as the existing class. This part of the rule first gets the relevant information from the annotations (using the labels assigned on the LHS of the rule), then adds the new information to the ontology and finally adds annotations to the entities in the document. NEBOnE is responsible also for ensuring that the resulting changes to the ontology are wellformed: this is described in more detail in Section 3.3. Figure 1 shows a screenshot from GATE of an ontology created from a selection of wikipedia texts about animals.

3.3 NEBOnE

The SPRAT application uses the specially developed NEBOnE plugin for GATE in order to generate the changes to the ontology. NEBOnE (Named Entity Based ONtology Editing) is an implementation for processing natural language text and

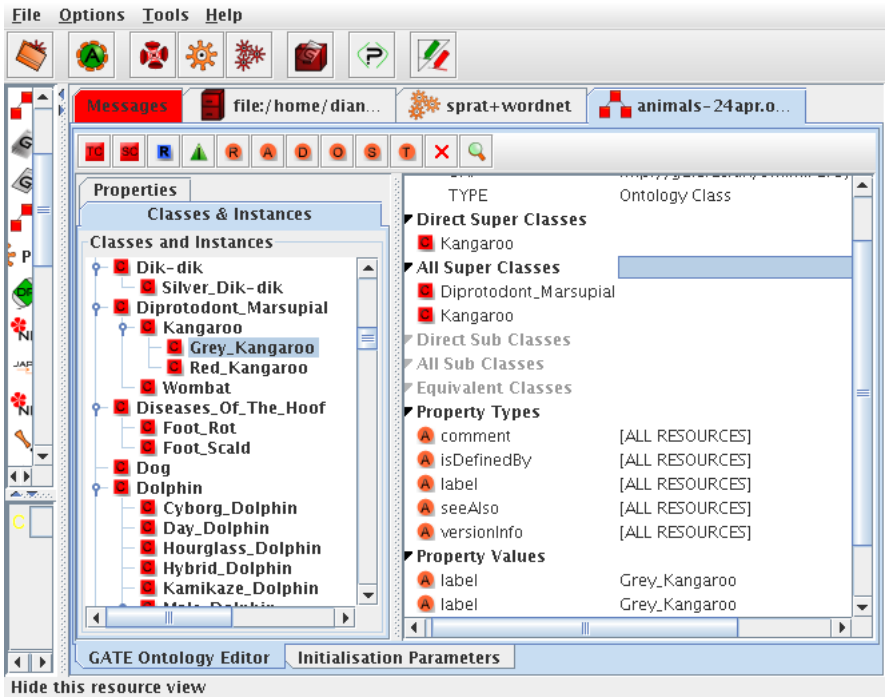

Fig. 1. Generated ontology in GATE

manipulating an ontology, and is derived from the CLOnE plugin [22] for GATE. The major difference between CLONE and NEBOnE is that while CLONE relies on a restricted input text (generated by the user in a controlled language), NEBOnE can be used with unrestricted free text, so it is a lot more flexible. When the NEBOnE plugin is installed, actions concerning the ontology are implemented on the right hand side of JAPE rules, such as adding or deleting new classes, instances, subclasses, properties and so on.

If an item is selected for addition to the ontology as a new class, NEBOnE first checks to see whether it already exists in the ontology: if it already exists in the place where it is scheduled to be added, NEBOnE will do nothing. If the item exists as a class elsewhere in the ontology, NEBOnE will add the new class (because it supports multiple inheritance). If the requested parent class and subclass both exist and are class names, NEBOnE will make the second a subclass of the first and print a message. If either is already an instance, or the parent class does not exist yet, NEBOnE will print a warning message.

Similarly, if an instance to be added already exists (as an instance), NEBOnE simply generates a notification message. If the item already exists as a class, and an instance of the same name is to be added, or vice versa, then NEBOnE will not generate the new instance and will produce a warning message. Thus NEBOnE

ensures consistency in the ontology, avoiding the need to run a checker after the ontology has been modified. A user can of course choose to ignore any potential inconsistencies, by checking the generated messages and then manually adding any offending items or making other changes to the ontology.

Unlike CLOnE, NEBOnE's functions will create classes and superclasses as required in order to accommodate instances and subclasses, respectively; it does not require every class to be explicitly created before it is used.

3.4 Evaluation

We evaluated the accuracy of the lexical patterns using a corpus of 25 randomly selected wikipedia articles about animals, such as the entries for *rabbit*, *sheep* etc. We ran SPRAT and examined the results in some detail. In total, SPRAT generated 1058 classes, of which 83.6% were correct; 659 subclasses, of which 76.6% were correct, 23 instances, of which 52.2% were correct, and 55 properties, of which 74.5% were correct. We should point out that in these type of texts (articles about animals) the number of instances is quite small. The wrongly extracted instances were largely the result of erroneous named entity recognition. Otherwise, we find the results very encouraging.

4 Ontology change management in GATE

The bottom-up approach to ontology change embodied by applications such as SPRAT is only half the story. The ontology development lifecycle also requires a top-down approach, embodied here by a change log management system which allows the managing of changes made by users in a collaborative environment [23]. This is necessary because different people, who may be experts in their domain, sometimes end up making changes to the same ontology, often at distributed locations. Where it is not possible to have access to a shared repository, other people wishing to make changes to the same ontology have to wait for others to finish their tasks. Changes made to an ontology may involve additions, deletions and modifications [24].

The basic idea behind producing a change log is to allow people to share their changes with others and thus collaborate on manipulating ontology data. This allows them to work independently but simultaneously on the same ontology. However, some ontologies can be extremely large and unwieldy, whereas the changes made to such ontologies may be only very minor. Instead of exchanging different versions of ontologies, it makes it easier to exchange change logs which can then be applied over the original ontology to get the modified version. The procedure of applying the change log is automatic, so no intervention is required by the user, who simply loads the old ontology and the changelog, which automatically updates the old ontology to the new one.

GATE has several plugins useful for carrying out different types of information extraction tasks. It also has its own ontology API, which can be used

together with these resources to take the maximum advantage of the combination. We implemented GATE Ontology Services (GOS) to allow people to connect to a central repository and manage ontologies centrally. It has been implemented so that it can be used with other resources available in GATE, such as the SPRAT application.

GOS supports storing ontologies in a shared repository on a remote server. GOS is based on OWLIM [25], which is a high performance semantic repository developed in Java. It has its own published API with which a user can make changes to the ontologies stored on the server. Software clients such as the *NeonOntologyServiceClient* allow users to connect to this server, upload new ontologies or use existing ones and manipulate data using the service methods published by the GOS. Similar to the NeOn Toolkit, GOS also produces a change log that describes changes made by the various users.

When making changes to the ontology, users contribute to the change log maintained on the server. The GOS maintains only one instance of change log per repository and accumulates changes made by different users in the same change log. Users wishing to download the entire change log can do so by selecting one of the options provided in the *NeonOntologyServiceClient*.

In its current version, GOS inherits Sesame's repository management system which allows giving different rights to different users on different repositories. If proper rights are assigned, multiple users can connect to the same repository from different locations and make changes. It is up to the system using OWLIM/Sesame as backend to utilise this functionality and restrict multiple users from editing the same ontology at the same time. In GOS, currently only one user is allowed to modify the ontology at a time. Other users are given read-only access if they connect to a repository which is already being edited by another user. However, since the different repositories are stored on the same central server, different users connecting to the same repository one after the other see the latest modified version. If simultaneous access is needed, one can carefully distribute or make copies of the same ontology across different repositories and ask users to connect to a different copy. Users wishing to make changes can then contribute by adding new instances or deleting existing ones in their copy. GOS does not allow making changes to the implicit resources (imported ontologies). This helps in restricting users from making changes to the basic taxonomy. More information on GOS, how to access it, its methods and the change log can be found in [26].

A typical scenario involves an ontology originating from an organisation such as FAO, a part of which is then modified in the NeOn Toolkit using the loosely coupled SPRAT plugin. Since SPRAT relies on the GATE architecture, the change log is produced by GATE as part of the ontology modification process. In order for the new version of the ontology to be then recoupled with the existing original ontology, and for further tools to be used on it, the change log must be readable also by the NeOn Toolkit so that the changes can be applied. We have tested the change log system in both directions, by verifying that changes made

to an ontology in GATE can be applied successfully to the original ontology in the NeOn toolkit, and vice versa, with identical ontologies created as a result.

5 Conclusions

We have described the implementation of the approach to modelling some of the dynamics of semantic metadata, providing mutual support for ontology lifecycle development and an NLP toolkit. We have developed a tool for deriving new semantic information from unstructured text and represent this information ontologically so that it may then be used for further processing. The importance of this work is the interoperability achieved between the ontology editing and natural language processing architectures. Users of both tools may decide to modify the ontologies in question or to update the source data or facts derived from them. By creating mechanisms for the interchange and propagation of information in this way, including valuable change log data and support for multiple users working individually or collaboratively in a potentially distributed environment with networked ontologies, we have released an important bottleneck.

Acknowledgements. This research was partially supported by the EU Sixth Framework Program project NeOn (IST-2005-027595).

References

1. Maynard, D., Li, Y., Peters, W.: NLP Techniques for Term Extraction and Ontology Population. In Buitelaar, P., Cimiano, P., eds.: Bridging the Gap between Text and Knowledge - Selected Contributions to Ontology Learning and Population from Text. IOS Press (2008)
2. Bontcheva, K., Davies, J., Duke, A., Glover, T., Kings, N., Thurlow, I.: Semantic Information Access. In Davies, J., Studer, R., Warren, P., eds.: Semantic Web Technologies. John Wiley and Sons (2006)
3. Noy, N., Sintek, M., Decker, S., Crubézy, M., Fergerson, R., Musen, M.: Creating Semantic Web Contents with Protégé-2000. IEEE Intelligent Systems **16**(2) (2001) 60–71
4. Harris, Z.: Mathematical Structures of Language. Wiley (Interscience), New York (1968)
5. Hirschman, L., Grishman, R., Sager, N.: Grammatically based automatic word class formation. Information Processing and Retrieval **11** (1975) 39–57
6. Maynard, D.G.: Term Recognition Using Combined Knowledge Sources. PhD thesis, Manchester Metropolitan University, UK (2000)
7. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Web-scale Information Extraction in KnowItAll. In: Proceedings of WWW-2004. (2004) `http://www.cs.washington.edu/research/knowitall/papers/www-paper.pdf`.
8. Pantel, P., Pennacchioni, M.: Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In: Proceedings of Conference on Computational Linguistics / Association for Computational Linguistics (COLING/ACL-06), Sydney, Australia (2006) 113–120

9. Berland, M., Charniak, E.: Finding parts in very large corpora. In: Proceedings of ACL-99, College Park, MD (1999) 57–64
10. Pantel, P., Ravichandran, D.: Automatically labeling semantic classes. In: Proceedings of HLT/NAACL-04), Boston, MA (2004) 321–328
11. Cimiano, P., Voelker, J.: Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery. In: Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB), Alicante, Spain (2005)
12. Cimiano, P., Hartung, M., Ratsch, E.: Learning the appropriate generalization level for relations extracted from the Genia corpus. In: Proc. of the 5th Language Resources and Evaluation Conference (LREC). (2006)
13. Gamallo, P., Gonzalez, M., Agustini, A., Lopes, G., de Lima, V.: Mapping syntactic dependencies onto semantic relations. In: Proc. of the ECAI Workshop on Machine Learning and Natural Language Processing for Ontology Engineering. (2006)
14. Maedche, A.: Ontology Learning for the Semantic Web. Kluwer Academic Publishers, Amsterdam (2002)
15. Fellbaum, C., ed.: WordNet - An Electronic Lexical Database. MIT Press (1998)
16. Schuler, K.K.: VerbNet: A broad-coverage, comprehensive verb lexicon. PhD thesis, University of Pennsylvania (2005)
17. Aswani, N., Tablan, V., Bontcheva, K., Cunningham, H.: Indexing and Querying Linguistic Metadata and Document Content. In: Proceedings of Fifth International Conference on Recent Advances in Natural Language Processing (RANLP2005), Borovets, Bulgaria (2005)
18. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Conference on Computational Linguistics (COLING'92), Nantes, France, Association for Computational Linguistics (1992)
19. de Cea, G.A., Gómez-Pérez, A., Ponsoda, E.M., Suárez-Figueroa, M.C.: Natural language-based approach for helping in the reuse of ontology design patterns. In: Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns (EKAW 2008), Acitrezza, Italy (September 2008)
20. Maynard, D., Funk, A., Peters, W.: SPRAT: a tool for automatic semantic pattern-based ontology population. In: International Conference for Digital Libraries and the Semantic Web, Trento, Italy (September 2009)
21. Cunningham, H., Maynard, D., Tablan, V.: JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield (November 2000)
22. Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., Handschuh, S.: CLOnE: Controlled Language for Ontology Editing. In: Proceedings of the 6th International Semantic Web Conference (ISWC 2007), Busan, Korea (November 2007)
23. Palma, R., Haase, P., Ji, Q.: Change management to support collaborative workflows. Technical Report D1.3.2, NeOn Project Deliverable (2009)
24. Maynard, D., Peters, W., D'Aquin, M., Sabou, M., Aswani, N.: Dynamics of metadata. Technical Report D1.5.1, NeOn Project Deliverable (2007)
25. Kiryakov, A.: OWLIM: balancing between scalable repository and light-weight reasoner. In: Proc. of WWW2006, Edinburgh, Scotland (2006)
26. Maynard, D., Peters, W., Angeletou, S., D'Aquin, M.: Implementation of metadata evolution. Technical Report D1.5.2, NeOn Project Deliverable (2008)