Using Lexico-Syntactic Ontology Design Patterns for ontology creation and population

Diana Maynard and Adam Funk and Wim Peters

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

Abstract. In this paper we discuss the use of information extraction techniques involving lexico-syntactic patterns to generate ontological information from unstructured text and either create a new ontology from scratch or augment an existing ontology with new entities. We refine the patterns using a term extraction tool and some semantic restrictions derived from WordNet and VerbNet, in order to prevent the overgeneration that occurs with the use of the Ontology Design Patterns for this purpose. We present two applications developed in GATE and available as plugins for the NeOn Toolkit: one for general use on all kinds of text, and one for specific use in the fisheries domain.

Key words: natural language processing, relation extraction, ontology generation, information extraction, Ontology Design Patterns

1 Introduction

Ontology population is a crucial part of knowledge base construction and maintenance that enables us to relate text to ontologies, providing on the one hand a customised ontology related to the data and domain with which we are concerned, and on the other hand a richer ontology which can be used for a variety of semantic web-related tasks such as knowledge management, information retrieval, question answering, semantic desktop applications, and so on.

Automatic ontology population is generally performed by means of some kind of ontology-based information extraction (OBIE) [1, 2]. This consists of identifying the key terms in the text (such as named entities and technical terms) and then relating them to concepts in the ontology. Typically, the core information extraction is carried out by linguistic pre-processing (tokenisation, POS tagging, etc.), followed by a named entity recognition component, such as a gazetteer and rule-based grammar or machine learning techniques.

In this paper we discuss the use of information extraction techniques involving lexico-syntactic patterns 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. While the use of such patterns is not new in itself (for example the Hearst patterns [3]), most previous work in this area has focused on using and extending or refining the set of Hearst patterns with additional information, or has focused on patterns within a very specific domain. In this work, we investigate the addition of lexico-syntactic patterns corresponding to ontology design patterns (ODPs) [4]which, in contrast to the Hearst patterns, are very general and thus cover a wide range of sentences but are also very ambiguous. While Hearst patterns produce high precision but low recall, the ontology design patterns produce high recall but low precision. We also include some additional lexico-syntactic patterns and investigate the addition of semantic restrictions to reduce the overgeneration problem. A detailed description of all the patterns and of the semantic restrictions is given in Section 2.

The system is implemented in GATE, an architecture for natural language processing which contains a number of pre-existing language processing components and applications, and enables the user to develop their own applications and integrate new plugins [5]. Two applications are available as plugins for the NeOn toolkit[1]: one for general use on all kinds of text, and one for specific use in the fisheries domain. These are described in more detail in Section 3.

2 Lexico-syntactic patterns

Traditional rule-based NE recognition applications usually rely on a fairly small set of patterns which aim to identify the relevant entities in text. These make extensive use of gazetteer lists which provide all or part of the entity in question, in combination with linguistic patterns (see for example [6] for a discussion of the importance of gazetteers in pattern-based NE recognition). A typical rule to identify a person's name consists of matching the first name of the person with an entry in the gazetteer (e.g. "John" is listed as a possible first name), followed by an unknown proper noun (e.g. "Smith", which is recognised as a proper name by the POS tagger). Most patterns include some combination of proper noun or word with an initial capital letter (for English) and either some gazetteer entry or linguistic feature.

However, identifying ontological concepts and/or relations requires a slightly different strategy. For open relation extraction [7], we have no such lists to use as a starting point. Even where we do have a seed ontology or known lists of terms and can make use of this information, it is often still insufficient because the concept may not be in the ontology yet, may be in the ontology but ambiguous, or may exist there in a different form (e.g. as a synonym or as a linguistic variant).

An alternative approach to traditional recognition techniques is to make use of linguistic patterns and contextual clues. Lexico-syntactic pattern-based ontology population has proven to be reasonably successful for a variety of tasks [8]. The idea of acquiring semantic information from texts dates back to the

[1] `http://www.neon-toolkit.org/`

early 1960s with Harris' *distributional hypothesis* [9] and Hirschman and Sager's work in the 1970s [10], 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 [11], 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 records. Here the structures are also quite simple, with short and relatively unambiguous sentences typically found: this makes syntactic pattern matching much easier.

We have identified three sets of patterns which can help us identify concepts, instances and properties to extend the ontology: the well-known Hearst patterns (Section 2.1), the Lexico-Syntactic Patterns developed in NeOn corresponding to Ontology Design Patterns (Section 2.2), and some new contextual patterns defined by us which take into account contextual information (Section 2.3). As a first step, we identified patterns which generated basic ontology elements such as instances, classes, subclasses and properties.

2.1 Hearst patterns

The Hearst patterns are a set of lexico-syntactic patterns that indicate hyponymic relations [3], and have been widely used by other researchers, e.g. [12]. Typically they achieve a very high level of precision, but quite low recall: in other words, they are very accurate but only cover a small subset of the possible patterns for finding hyponyms and hypernyms. The patterns can be described by the following rules, where NP stands for a Noun Phrase and the regular expression symbols have their usual meanings[2]:

- `such NP as (NP,)* (or|and) NP`
  Example: ... works by such *authors* as *Herrick*, *Goldsmith*, and *Shakespeare*.
- `NP (,NP)* (,)? (or|and) (other|another) NP`
  Example: *Bruises*, *wounds*, or other *injuries*...
- `NP (,)? (including|especially) (NP,)* (or|and) NP`
  Example: All *common-law countries*, including *Canada* and *England*...

Hearst actually defined five different patterns, but we have condensed some of them into a single rule. Also, where Hearst defines the relations as hyponym-hypernym, we need to be more specific when translating this to an ontology, as they could represent either instance-class or subclass-superclass relations. To make this distinction, we tested various methods. In principle, POS-tagging should be sufficient, since proper nouns generally indicate instances, but our tagger mistags capitalised common nouns (at the beginning of sentences) as proper nouns frequently enough that we cannot rely on it for this purpose. We also looked at the presence or absence of a determiner preceding the noun (since proper nouns in English rarely have determiners) and whether the noun is singular or plural, but this still left the problem of the sentence-initial nouns.

[2] `()` for grouping; `|` for disjunction; `*`, `+`, and `?` for iteration.

Finally, we decided to pre-process the text with the named entity recognition application ANNIE, and only consider certain types of named entities (*Person*, *Location*, *Organization*, and potentially some unknown entity types) as candidates for instances; all other NPs are considered to be classes. This gave much better results, occasionally missing an instance but rarely overgenerating.

2.2 ODP Lexico-Syntactic Patterns

The second type of patterns investigated was the set of Lexico-Syntactic Patterns (LSPs) corresponding to Ontology Design Patterns. We implemented a number of these patterns in our application. Some patterns could not be implemented because the GATE ontology API and the NEBONE plugin (which enables the ontology editing) do not contain the functionality for all restrictions.

In the following rules, `<sub>` and `<super>` are like variable names for the subclasses and superclasses to be generated; `CN` means *class of*, *group of*, etc.; `CATV` is a classification verb[3]; `PUNCT` is punctuation; `NPlist` is a conjoined list of NPs ("X, Y and Z").

1. Subclass rules
   - `NP<sub> be NP<super>`
   - `NPlist<sub> be CN NP<super>`
   - `NPlist<sub> (group (in|into|as) | (fall into) | (belong to)) [CN] NP<super>`
   - `NP<super> CATV CV? CN? PUNCT? NPlist<sub>`

   Example: *Frogs* and *toads* are kinds of *amphibian*.
   Thyroid medicines belong to the general group of *hormone medicines*.
2. Equivalence rules
   - `NP<class> be (the same as|equivalent to|equal to|like) NP<class>`
   - `NP<class> (call | denominate | (designate by|as) | name) NP<class>` (where the verb is passive)

   Example: *Poison dart frogs* are also called *poison arrow frogs*.
3. Properties
   - `NP<class> have NP<property>`
   - `NP<instance> have NP <property>`

   Example: *Birds* have *feathers*.
   Sharks have 32 teeth.

It is important to note that these particular LSPs were designed to be used as support for ontology modelling, rather than directly for automatic discovery of ontological relations. The effect of this is that while these patterns are quite productive (for example `X is a Y`), most of them are potentially ambiguous and susceptible to overgeneration when applied to the automatic discovery process. In particular, general patterns such as "NP have NP" and "NP be NP"

[3] E.g., *classify in/into*, *comprise*, *contain*, *compose (of)*, *group in/into*, *divide in/into*, *fall in/into*, *belong (to)*.

are very problematic. For example, the former pattern also matches sentences like "Writers have penguins based at the North Pole" and extracts the relation "writers have penguins" which is clearly wrong. Similarly, the pattern "NP be NP" would match sentences like "Sheep are a separate species" and extract the concept "sheep" as a subclass of "separate species" which makes no sense. There is also much ambiguity with this pattern: for example, in the sentence "Heliculture is the farming of snails", "heliculture" should be recognised as a synonym of "farming of snails" and not as a subclass. Clearly, such patterns are far too general to be used off the shelf. In Section 2.4 we discuss some restrictions we have implemented which aim to counteract these and other problems.

2.3 Contextual patterns

We also defined a set of rules designed to make use of contextual information in the text about known entities already existing in the ontology (unlike the lexico-syntactic patterns which assume no previous ontological information is present). These rules are used in conjunction with the OntoRootGazetteer plugin in GATE, which enables any morphological variant of any class, instance or label in the ontology to be matched with (any morphological variant of) any word or words in the text. Which elements from the ontology are to be considered (e.g., whether to include properties, and if so which ones) is determined in advance by the user when setting up the application. Note that because the generation process is incremental, involving a pipeline of sequential processsing resources, and because we use NEBOnE which generates the ontology on-the-fly, we do not necessarily need a seed ontology from which to start, because we can make use of the ontology entities already generated by the previous two sets of patterns. More information about the application and about NEBOnE is given in Sections 3 and 3.3 respectively.

Below we describe the contextual patterns we have identified:

1. **Add a new subclass**: `(Adj|N) NP<class>` → `NP<subclass>`.
   This matches a class name aready in the ontology preceded by an adjective or noun, such as adjective preceding a known type of fish, which we assume is a more specific type. For example, when we encounter the phrase . . . Japanese flounder. . . in a text and *flounder* is already in the ontology, we add *Japanese flounder* as a subclass of *flounder*.
2. **Add a new class** (a more generic version of the Hearst patterns). 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*.

3. **Add an alternative name as a synonym**: a name followed by an alternative name in brackets is a very common pattern in some kinds of text. For example in texts about flora and fauna we often get the common name followed by the Latin name in brackets, as in the following sentence:
   Example: *Mummichogs* (*Fundulus heteroclitus*) were the most common single prey item.
   If we know that one of the two NPs is a class or instance in the ontology, we can predict fairly accurately that the other NP is a synonym.

2.4 Adding semantic restrictions

Due to the overgeneralisation of some of the patterns described above, in particular the ODP LSPs, we have incorporated some restrictions on them. First, we restrict possible subclasses and classes to terms rather than to all NPs. For this, we use TermRaider, a term selection algorithm (currently unpublished) we have developed based on linguistic filtering and tf-idf scoring. This increases the precision dramatically, but lowers the recall a little; however, adjusting TermRaider's parameters to be a little more flexible with patterns should improve the recall.

The second restriction we imposed was to include lexical resources containing semantic classes from WordNet [13] and VerbNet [14], 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 [15], 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. We aim not only to reduce the number of errors, but also to eliminate the kind of general relations which while not incorrect, are not very useful. For example, knowing that a turtle is a local creature is not of much interest unless more contextual information is provided (i.e. in which region it is local).

Restrictions on Subclass Patterns One example of a restriction we placed was on the subclass rule `(Adj|N) NP<class>` → `NP<subclass>` from the set of contextual patterns, which we modified so that either the superclass must already exist in the ontology as a recognised class, or such that certain semantic restrictions apply. One such restriction states that both the proposed subclass and superclass must have the semantic category "animal". For example, this enables us to recognise "carrot weevil" as a subclass of "weevil". This rule in particular has very high accuracy (98%) and only seems to cause errors as a result of incorrect semantic categories from WordNet.

Restrictions on Properties One of the most error-prone rules was the Property rule `X has Y` from the Lexico-Syntactic Patterns set, which was clearly far too general. We restricted this to again use semantic categories of WordNet. For

patterns involving animals we can state that X must be an animal and Y must be a body part. This gave much better results (approximately 75% accuracy, although low recall). Another restriction is the type of thing that can be considered a property. We experimented with restricting the range of the property to the following semantic categories from WordNet: plant, shape, food, substance, object, body, animal, possession, phenomenon, artifact, and found much improved results.

3 SPRAT and SARDINE

We have developed two applications in GATE which make use of the lexico-syntactic pattern matching techniques to create and/or populate ontologies. Both applications are available as part of the GATE webservices (SAFE) plugin for the NeOn toolkit.

First, we have developed a generic application, SPRAT (Semantic Pattern Recognition and Annotation Tool) which can be used on any kind of text. This recognises new concepts, instances and properties, as described above, and adds these to a new or existing ontology. We have tested the application on wikipedia texts about animals (See Section 4) with good results so far, and plan to test on other domains and text types.

Second, we have developed a specific application, SARDINE (Species Annotation and Recognition and Indexing of Named Entities) which is aimed at the fisheries domain. The idea behind SARDINE is to identify mentions of fish species from text. The main difference between SARDINE and SPRAT is that, in addition to being developed for a specific domain, SARDINE also relies on a pre-existing domain-specific ontology which acts as a seed. We use the species ontology developed by the FAO[4] for this purpose. The application recognises:

- existing fish names listed in the seed ontology
- potential new fish names not listed in the seed ontology
- potential relations between fish names

For the new fish, it attempts to classify them in the ontology, based on linguistic information such as synonyms and hyponyms of existing fish. The application can either generate the new items directly into the seed ontology, or create a new ontology in the same way as SPRAT does. The latter is generally preferable because the original seed ontology is quite large and cumbersome, so it is easier to create a new smaller ontology which can then be easily verified by a human expert and then merged with the original seed ontology.

3.1 Processing Resources

Both applications are composed of a number of GATE components: some linguistic pre-processing followed by a set of gazetteer lists and the JAPE grammars described above. The components are as follows:

[4] Food and Agriculture Organization of the United Nations – `http://www.fao.org/`

– Tokeniser: divides the text into tokens
– Sentence Splitter: divides the text into sentences
– POS-Tagger: adds part-of-speech information to tokens
– Morphological Analyser: adds morphological information (root, lemma etc.) to tokens
– NP chunker: divides the text into noun phrase chunks
– Gazetteers: looks up various items in lists
– OntoRootGazetteer (optional): looks up items from the ontology and matches them with the text, based on root forms
– JAPE transducers: annotates text and adds new items to the ontology

The application can either create an ontology from scratch, or modify an existing ontology. SARDINE also requires the presence of a seed ontology, which could be the ontology to be modified, or a different one. The ontology used is the same one for the whole corpus: this means that if a number of documents are to be processed, the same ontology will be modified. If this is not the desired behaviour, then there are two options:

1. A separate corpus is created for each document or group of documents corresponding to a single output ontology. The application must be run separately for each corpus.
2. A processing resource can be added to the application that clears the ontology before re-running on the next document. This requires that the ontology is saved at the end of the application, after processing each document.

3.2 Implementation of patterns

The patterns themselves are implemented as JAPE rules [16]. On the left hand side (LHS) 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 pre-processing components (such as POS tagging, gazetteer lookup and so on) and (potentially) earlier JAPE rules. The example below shows a pattern for matching a subclass relation, such as "Frogs are a kind of amphibian" where "frog" is annotated as a subclass of "amphibian".

```
Rule:Subclass1
(
 ({NP}):sub
 ({Lookup.minorType == be}
 {Token.category == DT}{Lookup.majorType == kind})
 ({NP}):super
) --> ...
```

This pattern matches a noun phrase (identified by our NP Chunker), followed by some morphological variant of the verb "to be" (identified via the gazetteer lookup), a determiner (identified via the POS tagger), some word(s) indicating a "kind of" relation (identified via the gazetteer lookup) followed by another noun

phrase (identified by the NP Chunker). The two noun phrases (corresponding ultimately to the subclass and superclass) are given labels ("sub" and "super") which will used in the second part of the rule.

The right hand side (RHS) of the rule invokes NEBOnE and creates the new items in the ontology, as well as adding annotations to the document itself. 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. The RHS of the rule first gets the relevant information from the annotations (using the labels assigned on the LHS of the rule), then adds a new class below the root class for the superconcept (labelled "amphibian" in our example), a new subclass of this (labelled "frog" in our example), and finally adds annotations to the entities in the document. Figure 1 shows a screenshot from GATE of an ontology created.

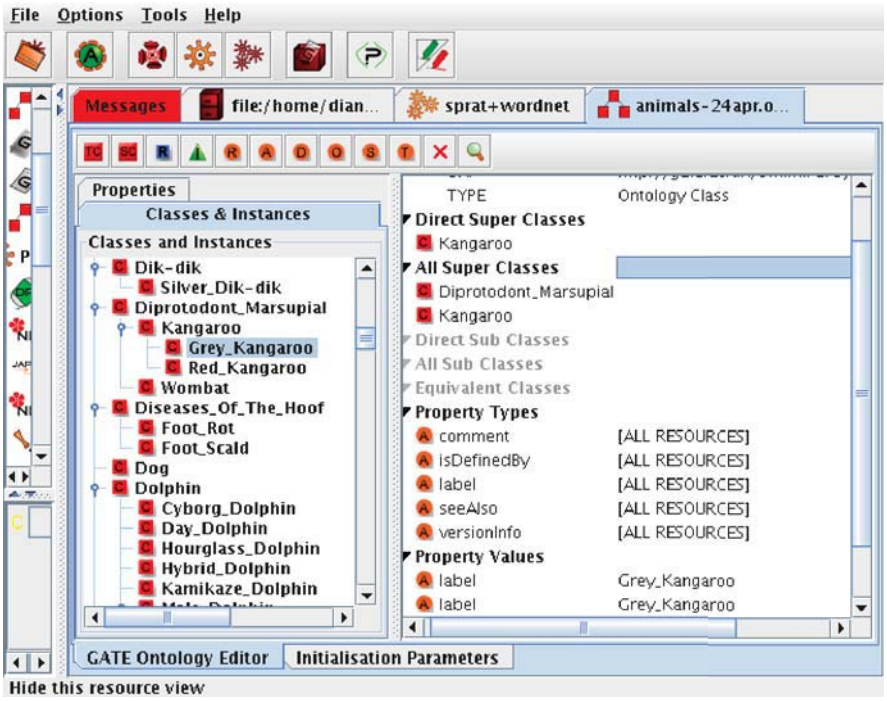

Fig. 1. Generated ontology in GATE

3.3 NEBOnE

Both applications use 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

manipulating an ontology, and is derived from the CLOnE plugin [17] 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 RHS (right-hand side) of JAPE rules, such as adding or deleting new classes, instances, subclasses, properties and so on.

Once the text has been pre-processed, a JAPE transducer processes each sentence in the input text and manipulates the ontology appropriately. This Processing Resource refers to the contents of the ontology in order to analyse the input sentences and check for errors; some syntactically identical sentences may have different results if they refer to existing classes, existing instances, or non-existent names, for example.

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[5]. 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. Note that, unlike in traditional named entity recognition evaluation, we use a strict method of scoring where a partially correct response, i.e. one where the span of the extracted entity is too short or too long, is considered as incorrect. This is because for ontology population, having an incorrect span is generally a more serious error than in named entity recognition.

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. For example, *Barbados Blackbelly* was wrongly recognised by the system as a named entity, and was therefore extracted as an instance rather than as a subclass of *Sheep Breed*.

While we find the initial results from SPRAT very encouraging, we can see that the patterns implemented are far from foolproof, since unlike with a controlled language such as CLOnE, we cannot rely on a one-to-one correspondence between a simple syntactic structure and its semantics. First we have the problem of overgeneration. Already, we have discarded some potential patterns (such as some of the ODP LSPs) that we consider to generate too many errors. Further refinement is still necessary here, either to remove other patterns or to reimplement them in a different way.

One of the main causes of overgeneration is caused by the span of the noun phrase describing the concept to be added to the ontology. We have experimented with different possibilities. A larger span provides finer distinctions and thus

[5] We have not currently evaluated SARDINE formally, but informal tests show similar results

better classes, but overgenerates considerably, while a smaller span produces more general classes but better accuracy (does not overgenerate so much). By restricting the noun phrases to terms recognised by TermRaider, we solve this problem somewhat, but this means that the results are only as good as the terms recognised. It is also apparent that sometimes the restriction to terms risks losing some important information. For example, in the sentence:

> Mygalomorph and Mesothelae spiders have two pairs of book lungs filled with haemolymph

we can correctly recognise the relation "*Mesothelae spiders* have *book lungs*", but a better relation might be "*Mesothelae spiders* have *two pairs of book lungs*". We might also want to capture the fact that the book lungs are filled with haemolymph.

Second, as we discussed earlier, and as mentioned in [4],lexico-syntactic patterns tend to be quite ambiguous as to which relations they indicate. For example, `NP have NP` could indicate an object property or a datatype property relationship. Also, English word order can lead to inverse relations. For example, in the sentence "A traditional Cornish pilchard dish is the stargazy pie", *stargazy pie* is a kind of Cornish pilchard dish, but the sentence can equally be written "The stargazy pie is a traditional Cornish pilchard dish". Here, the use of the definite and indefinite determiner helps to identify the correct relationship, but this is not always the case. Often, further context is also crucial. For example, in the sentence "Both African males and females have external tusks", it is not very useful to extract the concept *females* with the property *have external tusks* unless you know that *females* actually refers to female African elephants. To extract this information would require also coreference matching, which is planned for the future.

Finally, complex and negative sentences can cause errors. From the phrase "DAT is a legitimate therapy", we could easily deduce that *DAT* could be classified as an instance of *therapy*. However, further inspection of the wider context reveals that the opposite is true, as the sentence actually reads "...there is no compelling scientific evidence that DAT is a legitimate therapy." This is a common problem with shallow NLP systems.

Integration of a full parser has also been investigated, but discarded on the grounds of speed (full parsing is extremely computationally expensive in this situation). In particular, we found that the sentences in Wikipedia articles, which we have used for training and testing, are quite hard to parse well, because they frequently exhibit a long and complex sentence structure which is highly ambiguous to a parser. This causes not only speed but also accuracy problems.

5 Related work

As already mentioned, the use of lexico-syntactic patterns in itself is far from new and has already proved to be successful for a variety of tasks [8]. Various attempts have been made to extend the Hearst patterns in a semi-automatic

way, for example using the web as evidence [12]. Other methods focus mainly on a specific kind of pattern, such as *part-of* relations [18], or use clustering approaches [19]. 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 [20], which performs relation extraction on the basis of patterns. It combines machine learning approaches with basic linguistic processing such as tokenisation, lemmatisation and shallow parsing. Our approach differs in that it has a greater number of lexico-syntactic patterns, including the ODP ones, 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[6], 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, our approach 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, we do not make use of a parser. There are many applications that make use of syntactic dependencies, e.g. [21, 22]. 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 [23].

6 Conclusions and Further Work

In summary, the idea behind this work is to investigate the extent to which such patterns can be used either on their own or in conjunction with the user to generate or populate a more detailed ontology from text. Both SPRAT and SARDINE applications assist the user in the generation and/or population of ontologies from text. They are available to download for use as GATE Web-service plugins for the NeOn toolkit [7]. The lexico-syntactic patterns we have implemented provide a good basis, but there is some work still to go in improv-

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

[7] `http://www.neon-toolkit.org`

ing the rules, and we have put forward a number of suggestions for ways in which this might be done.

One further possibility for improvement is to incorporate combinations of Hearst patterns and statistically derived collocational information, because its combination with lexico-syntactic patterns has proven to improve precision and recall [24]. Integration of a full parser has also been investigated, but discarded on the grounds of speed (full parsing is extremely computationally expensive in this situation). In particular, we found that the sentences in Wikipedia articles, which we have used for training and testing, are quite hard to parse well, because they frequently exhibit a long and complex sentence structure which is highly ambiguous to a parser. This causes not only speed but also accuracy problems.

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

References

1. Maynard, D., Cunningham, H., Kourakis, A., Kokossis, A.: Ontology-Based Information Extraction in hTechSight. In: First European Semantic Web Symposium (ESWS 2004), Heraklion, Crete (2004)
2. Saggion, H., Funk, A., Maynard, D., Bontcheva, K.: Ontology-based information extraction for business applications. In: Proceedings of the 6th International Semantic Web Conference (ISWC 2007), Busan, Korea (November 2007)
3. 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)
4. 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)
5. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: 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 (ACL'02). (2002)
6. Mikheev, A., Moens, M., Grover, C.: Named Entity recognition without gazetteers. In: Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL'99). (1999) 1–8
7. Banko, M., Etzioni, O.: The tradeoffs between open and traditional relation extraction. In: Proceedings of ACL-08. (2008)
8. 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`.
9. Harris, Z.: Mathematical Structures of Language. Wiley (Interscience), New York (1968)
10. Hirschman, L., Grishman, R., Sager, N.: Grammatically based automatic word class formation. Information Processing and Retrieval **11** (1975) 39–57

11. Maynard, D.G.: Term Recognition Using Combined Knowledge Sources. PhD thesis, Manchester Metropolitan University, UK (2000)
12. 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
13. Fellbaum, C., ed.: WordNet - An Electronic Lexical Database. MIT Press (1998)
14. Schuler, K.K.: VerbNet: A broad-coverage, comprehensive verb lexicon. PhD thesis, University of Pennsylvania (2005)
15. 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)
16. 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)
17. 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)
18. Berland, M., Charniak, E.: Finding parts in very large corpora. In: Proceedings of ACL-99, College Park, MD (1999) 57–64
19. Pantel, P., Ravichandran, D.: Automatically labeling semantic classes. In: Proceedings of HLT/NAACL-04), Boston, MA (2004) 321–328
20. 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)
21. 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)
22. 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)
23. Maedche, A.: Ontology Learning for the Semantic Web. Kluwer Academic Publishers, Amsterdam (2002)
24. Cederberg, S., Widdows, D.: Using LSA and noun coordination information to improve the precision and recall of automatic hyponymy extraction. In: Proceedings of the 7th conference on Natural language learning at HLT-NAACL, Morristown, NJ (2003) 111–118