Flexible Ontology Population from Text: The OwlExporter

René Witte, Ninus Khamis, and Juergen Rilling

Department of Computer Science and Software Engineering
Concordia University, Montréal, Canada

Abstract

Ontology population from text is becoming increasingly important for NLP applications. Ontologies in OWL format provide for a standardized means of modeling, querying, and reasoning over large knowledge bases. Populated from natural language texts, they offer significant advantages over traditional export formats, such as plain XML. The development of text analysis systems has been greatly facilitated by modern NLP frameworks, such as the *General Architecture for Text Engineering* (GATE). However, ontology population is not currently supported by a standard component. We developed a GATE resource called the *OwlExporter* that allows to easily map existing NLP analysis pipelines to OWL ontologies, thereby allowing language engineers to create ontology population systems without requiring extensive knowledge of ontology APIs. A particular feature of our approach is the concurrent population and linking of a domain- and NLP-ontology, including NLP-specific features such as safe reasoning over coreference chains.

1. Introduction

Ontologies have become a major tool for developing semantically rich applications. Ontology models are capable of representing a large amount of information using a small number of axioms (individuals and relationships). The semantically rich models provide users with a high level conceptualization of the information, while at the same time enabling them to focus on specific parts of the model. Web ontologies developed in the OWL-DL (W3C, 2004) language also provide a standardized means for querying, linking, and reasoning about knowledge.

As a majority of the world's knowledge is encoded in natural language text, automating the population of these ontologies using results obtained from NLP analysis of documents (Cimiano, 2006) has recently become a major challenge for NLP applications (Figure 1).

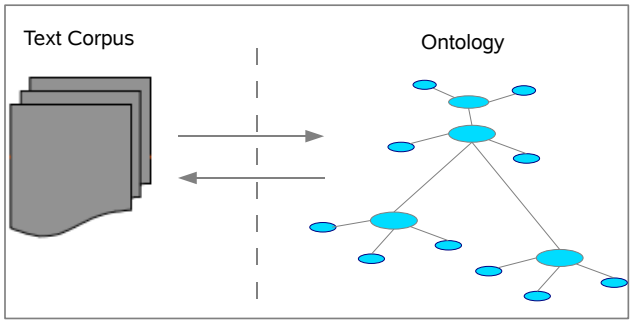

Figure 1: Populating Ontologies from Text

NLP application development has been greatly facilitated by integrated frameworks such as GATE (Bontcheva et al., 2004) or UIMA (Ferrucci and Lally, 2004). Standard tasks such as tokenization, POS tagging, or chunking are supported by a large number of existing components that can be easily assembled into more complex application pipelines. However, exporting the results of these NLP analyses into an ontology (ontology population) still requires large manual efforts on part of the language engineer.

We have implemented a GATE processing resource called the *OwlExporter* that allows to largely automate ontology population from text for an existing application pipeline. In addition to domain-specific concepts, it provides a number of novel features that address the particularities of NLP processing, such as exporting sentences, noun and verb phrase chunks, and integrating reasoning support for coreference chains. The component, as well as supporting resources, is available under an open source license.[1]

2. Motivation

The GATE NLP framework (Bontcheva et al., 2004) already provides means for working with ontologies using an integrated ontology layer. However, our component significantly enhances the state of the art by four important aspects:

Support for OWL-DL. GATE's ontology layer is based on OWLIM (Kiryakov et al., 2005), which only supports the less semantically rich OWL-Lite language. Our component enables populating OWL-DL ontologies, which is important for users that need to create more expressive models, and benefit from the inferences created by a description logic reasoner such as Racer (Haarslev and Möller, 2001), Pellet (Sirin et al., 2007), or FaCT++ (Tsarkov and Horrocks, 2006).

Automated Ontology Export. While a standard task, populating ontologies is not supported through a standard component. Population of ontologies needs to be performed using GATE's ontology API by writing additional custom code in Java. Language engineers need to have a good knowledge of the OWL formalism, such as how to model semantically rich axioms to create usable ontologies.

In contrast, our OwlExporter is a component that provides a separation of concerns, where ontology population is largely automated and can easily be added by a language engineer without requiring extensive knowledge of programming or ontology languages.

NLP Ontology Population. NLP pipelines typically build annotations during processing that are not directly related to the application domain, but can provide important information during subsequent analysis. For example, a bio-NLP pipeline would discover information about organisms,

[1] OwlExporter, http://www.semanticsoftware.info/owlexporter

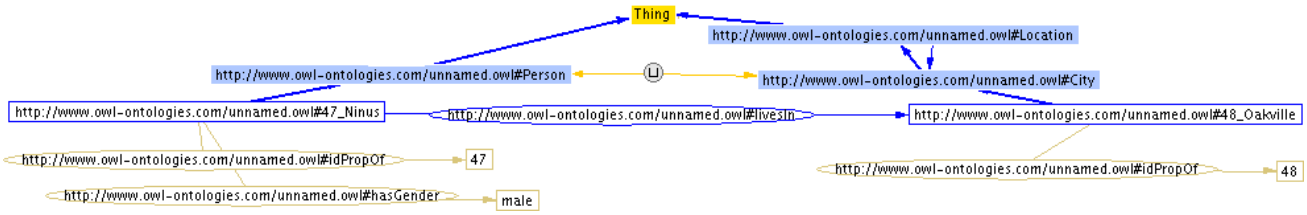

Figure 2: Exported Domain Entities of a Corpus

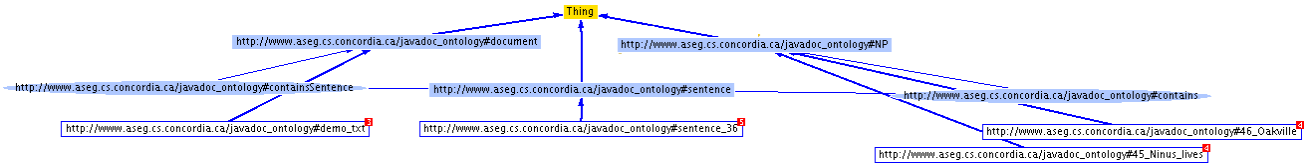

Figure 3: Exported NLP Entities of a Corpus

drugs, proteins, etc., but also create structural information like sentences, noun phrase chunks, or coreference chains. We provide automated support for integrating these information into the populated ontology and create the necessary relations for safe reasoning on them.

Addressing these three factors enables the OwlExporter to provide an automated, portable and simplified means of representing the knowledge found in text as instances and relationships within an ontology.

3. Related Work

We are unaware of any components designed to run within GATE that provide generic ontology population support for existing NLP pipelines. The efforts of (Chiarcos et al., 2008) aims at 1) finding a common ground between *annotations* and *features* created by different NLP tools 2) developing a generic XML schema to represent the *annotations* 3) store the marked up information in a database, and finally 2) use a linguistic ontology to represent the POS and GLOSS *annotations*. The tool however is fully automated with limited control from the user over what gets processed. Since the information found in text is completely arbitrary, introducing a fully automated tool to process such information will prove expensive. The OwlExporter gives the user full control over what domain specific and NLP information from the corpus get exported into the ontology.

The efforts of (Java et al., 2007) were geared towards creating OWL models from an existing "NLP-Oriented" (Java et al., 2007) ontology called OntoSem using an extension called OntoSem2OWL. Because OntoSem2OWL is tied to a single source, the validity of the OWL models produced by OntoSem2OWL rely solely on OntoSem.

Hayashi (Hayashi, 2007) proposed a taxonomic classification of the various processing and language resources within GATE. The works however does not attempt to implement an automated means of populating the taxonomy. Klein and Potter (Klein and Potter, 2004) also attempted an ontology for NLP services, however; their work did not include "detailed taxonomies" as described in (Hayashi, 2007).

4. Design and Implementation

GATE applications are assembled from individual components, which all add information to the documents in a corpus. For example, part-of-speech tags, noun phrase chunks, or named entities are all recorded in individual *annotations* attached to a document.

The core idea of our OwlExporter is to take the annotations generated by an NLP pipeline and provide for a simple means of establishing a mapping between NLP and domain annotations on one hand and the *concepts* and *relations* of an existing NLP and domain-specific ontology on the other hand. The former can then be automatically exported to the ontology in form of *individuals* and the latter as *datatype* or *object properties* (Figure 2).

The resulting, populated ontology can then be used within any ontology-enabled tool for further querying, reasoning, visualization, or other processing (Figure 4).

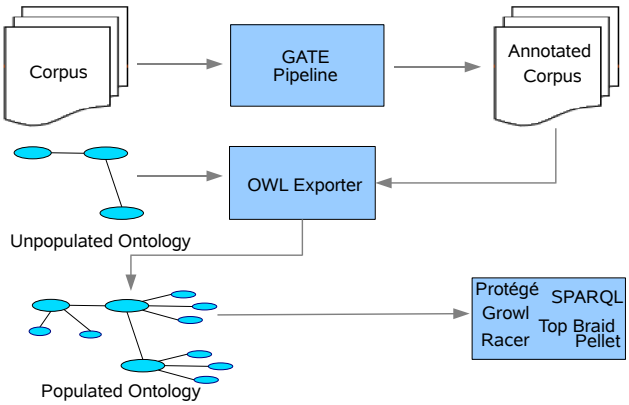

Figure 4: General Workflow of the OwlExporter

The OwlExporter is able to populate already existing domain specific ontologies in any OWL format, as well as NLP ontologies like the GOLD Ontology (Farrar and Langendoen, 2003) that model commonly used concepts in the language engineering domain (Figure 3).

4.1. Mapping NLP Annotations to OWL Concepts

The language engineer only needs to add two simple grammar rules to allow ontology population from an existing application. In essence, two new annotations need to be created that declare the mapping between the NLP annotations (created during processing) and the external NLP and domain ontologies (created by an ontology engineer):

OwlExportClass: This annotation records which document annotations need to be exported, and to which ontology class. For example, an NE system might recognize entities such as a "Person" or "Company." The `OwlExportClass` annotation could then specify to export Person → Person and Company → Organization, i.e., each detected person will become an instance of the ontology class Person and each company NE an instance of Organization.

OwlExportRelation: This annotation defines the export of relations between entities, which are recorded using OWL *object properties*. For example, an IE system might additionally detect that a certain person *works_for* a certain company. To export this as an ontology relation between the Person and Organization instances, each such relation is mapped using an `OwlExportRelation` annotation.

For entities in a text corpus that need to be exported as datatype relationships, an additional feature is required in the `OwlExportClass` annotation that matches the name of the property within the ontology (for example, `hasGender=Male`). Our OwlExporter handles subsumption required for attribute export; For example, consider the following KB: Person, Male ⊏ Person, Female ⊏ Person, hasName(Person,`xsd:string`). If we attempt to export a Female name, the OwlExporter is able to process this expression, knowing that *hasName* (Person, `xsd:string`) also applies to *sub-concepts* of Person.

These rules can be easily written in GATE's JAPE language (Cunningham et al., 2000). In Figure 5 we show an example of a JAPE grammar that accepts the `Mention` annotation as input, and creates the new annotation `OwlExportClass` with the list of features such as `kind, className` and `corefchain`.

4.2. Domain and NLP Ontologies

As discussed above, the OwlExporter uses two ontologies for populating a text corpus. The first ontology is a domain specific ontology that models concepts and relationships that are relevant to the given domain (e.g., biology, architecture, software engineering); and the second ontology is a domain independent NLP ontology that contains concepts commonly used in language engineering. The NLP ontology can contain concepts such as Document, Sentence, NP and VP and the OwlExporter automatically populates the NLP ontology with relationships such as hasEntity or containsSentence that associate the individuals from the domain ontology to the individuals of the NLP ontology, as shown in Figure 6. This provides for advanced queries and reasoning on the populated ontology, e.g., finding all entities of a certain concept that appear in the same sentence as another object, or which NPs exist in a given sentence.

```
Rule: mention_owl_class
(
    {Mention}
):ann
-->
{
    try {
        AnnotationSet as =
                (gate.AnnotationSet)bindings.get("ann");
        FeatureMap features =
            Factory.newFeatureMap();

        for(Annotation ann : as) {
            String instanceName = doc.getContent().getContent(
                ann.getStartNode().getOffset(),
                ann.getEndNode().getOffset()).toString();
            features.put("kind",ann.getFeatures().get("kind"));
            features.put("className",ann.getFeatures().get("class"));
            features.put("representationId",ann.getId());
            features.put("id",ann.getId());
            features.put("corefchain", null);
            features.put("instanceName", instanceName);
        }

        outputAS.add(as.firstNode(), as.lastNode(),
                    "OwlExportClass",features);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

Figure 5: An Example of an OwlExportClass JAPE Grammar

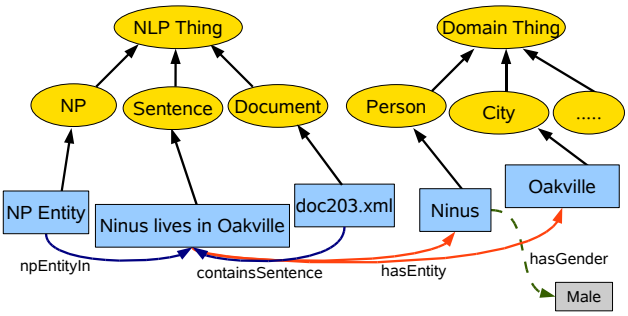

Figure 6: NLP and Domain Ontology Export Example

4.3. Coreference Chain Export and Reasoning

The OwlExporter also support modelling entities that reappear in different parts of a corpus, and that are linked together using *coreference chains*. For example, an NE coreferencer such as the one in ANNIE (Cunningham et al., 2002) can identify the nominal and pronominal coreferences between entities and create chains that link them together. Our OwlExporter can use the identified coreferences to semantically enrich the populated ontology, so that a reasoner will be aware of them.

When the coreferencer identifies entities of a corpus as being part of the same referent or representative an annotation (for example, `NP Chain`) is created that contains a list of entities that make up a coreference chain. The OwlExporter's `OwlExportClass` annotation accepts a `corefChain` feature that contains the ID of the *coreference chain* annotation. With the simple inclusion of the `corefChain` feature (a run-time option) the OwlExporter:

Creates the corefSentenceWithId relationship: The corefSentenceWithId relationship associates the

Type	Start	End	Id	Features
Person	0	5	11	gender=male, rule=FirstName
OwlExportClass	0	5	15	representationId=11, Kind=Class, className=Person, instanceName="Ninus", hasGender=male
Location	15	23	12	locType=city, rule=Location
OwlExportClass	0	5	15	representationId=12, Kind=Class, className=City, instanceName="Oakville"

Table 1: Mapping existing ANNIE annotations to OwlExportClass annotation

Type	Start	End	Id	Features
Class	30	45	7991	class=Class, kind=Class, instanceName=FeatureRenderer
Interface	52	60	7992	class=Class, kind=Class, instanceName=Renderer
OwlExportRelation	3638	3651	11176	domainId=7991, propertyName=hasSuperClass rangeId=7992, Kind=Relation

Table 2: Mapping existing relationship annotations to the OwlExportRelation annotation

referent in a chain with the sentences containing the occurrences of the coreferences.

Creates the corefStringWithId relationship: The corefStringWithId relationship ties the referent to it the multiple occurrences of its coreferences.

Creates the sameAs relationship: The OwlExporter establishes the links between the individuals in the ontology using the symmetric owl:sameAs property (Franz Baader et al., 2007). This allows the related individuals to be classified by an OWL reasoner as being equivalent.

In Figure 7 we show the *corefSentenceWithId* and *corefStringWithId* relationships created by the OwlExporter for a coreference chain. Figure 10 shows how the OwlExporter models coreference chains using the owl:sameAs property.

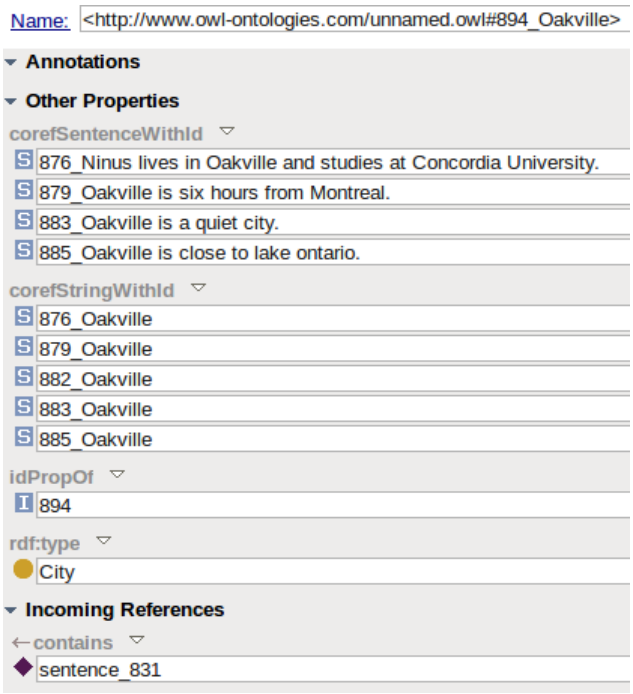

Figure 7: Coreference Chain Relationships exported into an OWL Ontology

Implementation. For implementing the OwlExporter we use the Protégé 4 API for tasks such as creating concepts, individuals, and relationships. We also use the GATE 5.1 API for processes such getting the values of run-time parameters, extracting content from the text corpus and getting the annotation sets.

5. Application and Evaluation

In this section we briefly illustrate how the OwlExporter is being used within different NLP applications, and benchmark how long it takes the OwlExporter to populate ontologies using corpora of different sizes.

5.1. General Workflow

Populating ontologies using the entities in a text corpus and the OwlExporter is achieved by a few simple steps:

1. Creating a new or reusing an existing domain ontology in association with a generic NLP ontology.
2. Using a GATE pipeline with its set of processing resources, and the annotations created by it.
3. Creating the mappings between the pipeline's existing annotations and the annotations needed by the OwlExporter:
   (a) Map the pipeline's annotations that are to be exported as instances of the ontology to the `OwlExportClass` annotation.
   (b) Map the annotations that are to be exported as relationships in an ontology to the `OwlExportRelation` annotation.
4. (optional) Mapping annotations created by a coreferencer to the `OwlExportClass` annotation using the name of the *coreference chain annotation*, and the `corefChain` *feature*.

5.2. Application Examples

In this section, we show how some existing applications make use of the OwlExporter.

Information Extraction. ANNIE is an example information extraction system distributed with the standard GATE distribution (Cunningham et al., 2002). Using our OwlExporter, we can easily extend it into an ontology population system.
Table 1 shows some example annotations created by the ANNIE pipeline, followed by the `OwlExportClass` annotations needed by the OwlExporter to export individuals to the ontology. In this example, we use an already existing

GATE Application	Number of Documents	Corpus Size (words)	Number of Exported Individuals	Number of Exported Relationships	Export Duration (sec)
Software Miner	4	40,727	239	653	6.21
Organism Tagger	19	226,397	609	6,642	38.73
Durm Architecture	1	215,097	6791	416,936	1181.94

Table 3: OwlExporter Benchmarks

ontology "demo.owl"[2] that models commonly used concepts in GATE such as Person, Place and Organization.
In Figure 8 we show the list of annotations created by the algorithmic processing resources of the ANNIE pipeline.

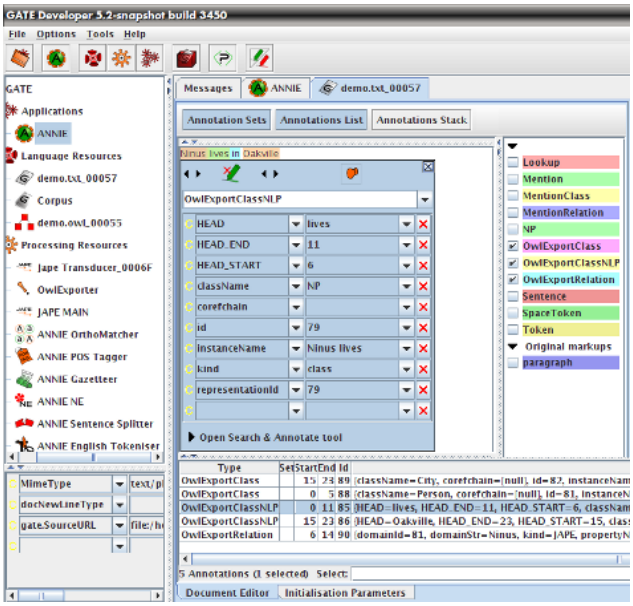

Figure 8: Annotations created by the ANNIE Pipeline and Annotation Features used for OWL export

The *annotation set* includes the OwlExportClass and OwlExportClassNLP *annotations* that are exported to the ontology. Also shown are the features that belong to the OwlExportClassNLP *annotation* for an NP entity in the corpus.
In Figure 2 we show how entities of a corpus that have been tagged as being Person, and Location are exported to the domain ontology. And finally in Figure 3 we show how the Document, Sentence, and NP entities from a corpus are exported as *instances* of their related concepts in the NLP ontology.

Software Engineering. In a software text mining application (Witte et al., 2007), we detect relationships between domain-specific concepts such as software *classes* and *interfaces*. Exporting the NLP-detected relations, extracted from software documentation, into an ontology allows software engineers to run semantically-oriented queries on the populated ontology to obtain information for software maintenance tasks (Witte et al., 2007). In Table 2 we show an example of annotations created by the software miner pipeline, followed by the `OwlExportRelation` annotation needed by the OwlExporter to export relationships to the ontology.

[2]demo.owl is bundled with GATE and can be accessed from the `GATE_HOME/plugins/Ontology/Tools/resources` directory.

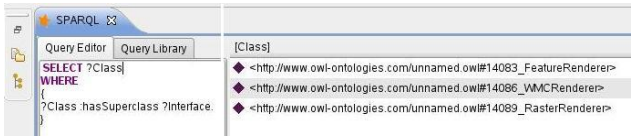

Figure 9: SPARQL Queries on populated Software Ontology

Figure 9 shows a SPARQL query applied to the populated Software Miner ontology that extracts all the individuals that are of type Class and that are specializations of a given Interface.

BioNLP. In this example from a bio-NLP application (Witte and Baker, 2005) we illustrate the export of a coreference chain into an ontology.

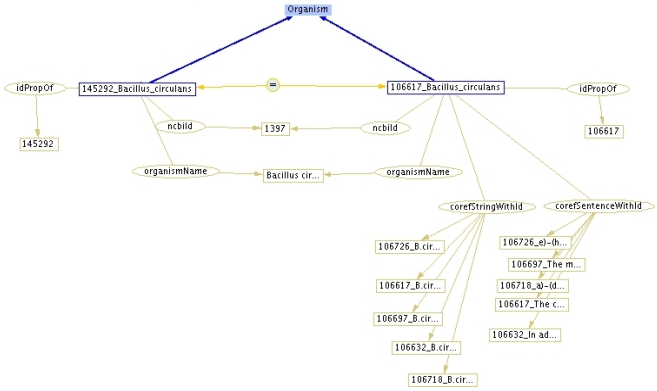

Figure 10: Entities in a Coreference Chain linked together using the owlSameAs property

Figure 10 shows an excerpt of the populated ontology; note the "=" node indicating the equivalence between the two Organisms *Bacillus circulans* detected in a text, together with their individual occurrences in sentences (corefSentenceWithId) and individuals (corefStringWithId).

5.3. OwlExporter Evaluation

In Table 3 we have summarized a set benchmarks that measure how long it takes the OwlExporter to export annotations. We have included the size of the text corpus, the number of created individuals and created relations. As can be seen from the table, a large number of entities can be exported without significant performance penalties.

6. Conclusion and Future Work

Within this work, we emphasized on the importance of the Web Ontology Language (OWL) in language engineering. We stressed the need of a simple and efficient solution that enables the population of ontologies within existing NLP frameworks, in particular GATE.

We introduced the OwlExporter processing resource, a tool that is designed to automate the process of ontology export from text. It has already been used for a number of real-world applications in various domains, and is now made generally available under an open source license.

Acknowledgements. Qiangqiang Li and Thomas Kappler contributed to the design and implementation of the OwlExporter.

7. References

K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. 2004. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10(3/4):349—373.

Christian Chiarcos, Stefanie Dipper, Michael Götze, Ulf Leser, Anke Lüdeling, Julia Ritz, and Manfred Stede. 2008. A Flexible Framework for Integrating Annotations from Different Tools and Tagsets. In *Traitement Automatique des Languages*.

Philipp Cimiano. 2006. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

H. Cunningham, D. Maynard, and V. Tablan. 2000. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, November.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: an Architecture for Development of Robust HLT Applications. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL)*.

Scott Farrar and Terry Langendoen. 2003. A Linguistic Ontology for the Semantic Web. *GLOT International*, Volume 7.

David Ferrucci and Adam Lally. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348.

Diego Calvanese Franz Baader, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

V. Haarslev and R. Möller. 2001. Racer system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, Siena, Italy*, pages 701–705. Springer-Verlag.

Yoshihiko Hayashi. 2007. A linguistic service ontology for language infrastructures. In *ACL '07: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 145–148, Morristown, NJ, USA. Association for Computational Linguistics.

Akshay Java, Sergei Nirenburg, Marjorie McShane, Tim Finin, Jesse English, and Anupam Joshi. 2007. Using a Natural Language Understanding System to Generate Semantic Web Content. *International Journal on Semantic Web and Information Systems*, 3(4), November.

Atanas Kiryakov, Damyan Ognyanov, and Dimitar Manov. 2005. OWLIM - A Pragmatic Semantic Repository for OWL. In Mike Dean, Yuanbo Guo, Woochun Jun, Roland Kaschek, Shonali Krishnaswamy, Zhengxiang Pan, and Quan Z. Sheng, editors, *WISE Workshops*, volume 3807 of *Lecture Notes in Computer Science*, pages 182–192. Springer.

E. Klein and S. Potter. 2004. An Ontology for NLP Services. In *LREC '04: Proceedings of Workshop on a Registry of Linguistic Data Categories within an Integrated Language Resource Repository Area*, May.

E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, and Y. Katz. 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53, June.

Dmitry Tsarkov and Ian Horrocks. 2006. FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006*, pages 292–297. Springer.

W3C. 2004. OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/.

René Witte and Christopher J.O. Baker. 2005. Combining Biological Databases and Text Mining to support New Bioinformatics Applications. In *Natural Language Processing and Information Systems: 10th International Conference on Applications of Natural Language to Information Systems (NLDB 2005)*, volume 3513 of *LNCS*, pages 310–321, Alicante, Spain, June 15–17. Springer-Verlag. http://dx.doi.org/10.1007/11428817_28.

René Witte, Yonggang Zhang, and Juergen Rilling. 2007. Empowering Software Maintainers with Semantic Web Technologies. In E. Franconi, M. Kifer, and W. May, editors, *4th European Semantic Web Conference (ESWC 2007)*, number 4519 in LNCS, pages 37–52, Innsbruck, Austria, June 3–7. Springer-Verlag Berlin Heidelberg. http://www.eswc2007.org/pdf/eswc07-witte.pdf.