D2.3.6 Prototypes of language dependent tools for evaluation

Diana Maynard (University of Sheffield)

with contributions from:
Wim Peters (University of Sheffield)
Marta Sabou (Vrije Universiteit Amsterdam)
Philipp Cimiano (University of Karlsruhe)

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB
Deliverable D2.3.6 (WP2.3)

This report accompanies the software prototypes of some language-independent tools for ontology evaluation. It describes briefly the tools and gives instructions for installation and use. There are two categories of tools presented: a set of tools for visualisation of generated ontologies and their evaluation results, and a set of tools for evaluating generated and populated ontologies using different metrics.
Keyword list: keywords useful for document classification and information retrieval

Document Identifier	KWEB/2005/D2.3.6/v1.0
Project	KWEB EU-IST-2004-507482
Version	v1.0
Date	November 30, 2005
State	final
Distribution	public

Knowledge Web Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2004-507482.

University of Innsbruck (UIBK) - Coordinator
Institute of Computer Science
Technikerstrasse 13
A-6020 Innsbruck
Austria
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

École Polytechnique Fédérale de Lausanne (EPFL)
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

France Telecom (FT)
4 Rue du Clos Courtel
35512 Cesson Sévigné
France. PO Box 91226
Contact person : Alain Leger
E-mail address: alain.leger@rd.francetelecom.com

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

Free University of Bozen-Bolzano (FUB)
Piazza Domenicani 3
39100 Bolzano
Italy
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

Institut National de Recherche en Informatique et en Automatique (INRIA)
ZIRST - 655 avenue de l'Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

Centre for Research and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH)
1st km Thermi - Panorama road
57001 Thermi-Thessaloniki
Greece. Po Box 361
Contact person: Michael G. Strintzis
E-mail address: strintzi@iti.gr

Learning Lab Lower Saxony (L3S)
Expo Plaza 1
30539 Hannover
Germany
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

National University of Ireland Galway (NUIG)
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Contact person: Christoph Bussler
E-mail address: chris.bussler@deri.ie

The Open University (OU)
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
Contact person: Enrico Motta
E-mail address: e.motta@open.ac.uk

Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

University of Karlsruhe (UKARL)
Institut für Angewandte Informatik und Formale Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Contact person: Rudi Studer
E-mail address: studer@aifb.uni-karlsruhe.de

University of Liverpool (UniLiv)
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
Contact person: Michael Wooldridge
E-mail address: M.J.Wooldridge@csc.liv.ac.uk

University of Manchester (UoM)
Room 2.32. Kilburn Building, Department of Computer Science, University of Manchester, Oxford Road
Manchester, M13 9PL
United Kingdom
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

University of Sheffield (USFD)
Regent Court, 211 Portobello street
S14DP Sheffield
United Kingdom
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

University of Trento (UniTn)
Via Sommarive 14
38050 Trento
Italy
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

Vrije Universiteit Amsterdam (VUA)
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
Contact person: Frank van Harmelen
E-mail address: Frank.van.Harmelen@cs.vu.nl

Vrije Universiteit Brussel (VUB)
Pleinlaan 2, Building G10
1050 Brussels
Belgium
Contact person: Robert Meersman
E-mail address: robert.meersman@vub.ac.be

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to writing parts of this document:

University of Manchester
University of Sheffield
Vrije Universiteit Amsterdam
University of Karlsruhe

Executive Summary

This report accompanies the software prototypes of language-dependent tools for ontology evaluation. It describes briefly the tools and how to acquire, install and use them. There are two categories of tools presented: first, a set of visualisation tools which enables the expert user to view generated ontologies and to perform his own evaluation on them; second, a set of tools enabling the user to evaluate generated and populated ontologies using different metrics. Further details and comparisons of such metrics will be described and discussed in future deliverables.

Contents

Chapter 1

Introduction

In the field of bioinformatics, there has been increasing interest in the use of ontologies, because they provide a means of accessing the information stored in large databases not only by humans (as traditionally was the case) but also by computers. The Gene Ontology (GO)[1] is one of the largest and most important ontologies in the field. By storing terms and their relations and thereby providing a standard vocabulary across many different resources, it enables annotation and querying of databases such as SWISS-PROT. For example, Lord et al. [PSBC03] present methods for measuring semantic similarity in GO in order to enable querying of such databases for e.g. proteins semantically similar to a query protein (i.e. which may have different names but are essentially the same).

In the last decade, methods for information extraction have become extremely important in the field of biomedicine. Detecting gene and protein names in texts is essential for knowledge discovery, and the lack of standardisation and presence of both ambiguous terms and term variation makes the task very complex. Similar mechanisms to carry out such tasks have been used as for traditional open-domain information extraction (i.e. both rule-based and machine learning approaches); however, the increasing use of ontologies has paved the way for the application of ontology-based information extraction techniques in this domain [WLT$^+$04, BTMC04]. The development of such applications is hampered by the lack of standardisation and suitable metrics for testing and evaluation [May05].

Until now, ontologies in biology were considered as mere guides for data structure, with the main purpose being to access the most useful documents and articles according to the researcher's interests. Applications such as semantic annotation enable us to combine and associate existing ontologies in the biological field, and to perform an integral modelling of the disparate biological data sources. Once this is achieved, knowledge can be extracted from the data repositories by means of agents, annotations and the semantic grid. Semantic annotation and ontology-based information extraction technologies form the cornerstone of text mining applications for the Semantic Web in fields such as bioinformatics. There has been a great deal of work in the last decade on evaluating traditional

[1]http://www.geneontology.org

information extraction (IE) systems in terms of performance, e.g. [Chi92], but there has until recently been a dearth of work on evaluating ontology-based applications such as automatic ontology generation from text and ontology-based information extraction or ontology population.

This deliverable was partly motivated by the problem of versioning the Gene Ontology (GO) and by similar cases in the bioinformatics domain: these are clear cases where methods for ontology evaluation are important, particularly given the extremely complex structure of such ontologies. At a recent All Hands Meeting hosted by the UK E-Science Programme (AHM 2005)[2], this issue was discussed by many researchers in bioinformatics and there was much interest in such evaluation tools.

However, the work presented here is designed to be generic and can be used in any kind of domain or application. In particular, it forms a part of the ontology lifecycle presented in this Work Package. This lifecycle consists of the following stages: Generation - Versioning - Evaluation / Visualisation - Negotiation. In particular, we see the evaluation and visualisation stage as complementary to the generation and versioning, as it enables the developer to visualise the effect his changes have made and it aids him to perform analysis, for example by showing differences in the attachment of instances to concepts. The evaluation of the concept hierarchy also helps to highlight particular changes and to show how this reflects on the ontology overall. The visualisation idea used in this deliverable can also be extended to show different versions of ontologies at the concept level.

This report accompanies the deliverable D2.3.6 ”Prototypes of language-dependent tools for ontology evaluation”. It describes briefly the tools and how to acquire, install and use them. There are two categories of tools presented: first, a set of visualisation tools which enables the expert user to view generated ontologies and to perform his own evaluation on them; second, a set of (non-graphical) tools enabling the user to evaluate generated and populated ontologies using different metrics. Further details and comparisons of such metrics will be described and discussed in future deliverables.

[2]http://www.allhands.org.uk

Chapter 2

Visualisation Tools for Ontology Evaluation

2.1 Introduction

This deliverable contains 3 applications for visualising generated ontologies in GATE:

1. A visual output for evaluation (CorpusAnnotationComparison);
2. A visual presentation of the extracted ontology (OntologyBuilderDisplay);
3. A visual representation of shared concepts in the case of multiple document collections (OntologyBuilderSourceDisplay).

They are available for download as a zipped package from
http://gate.ac.uk/projects/knowledge-web/tools.html

2.2 Required software

The applications for visualisation have been integrated in the GATE architecture[CMBT02], which is freely available for research purposes and can be downloaded from http://gate.ac.uk/download. Please note that the current version MUST be used with GATE version 3.0 and not GATE version 3.1: future versions of the software will be compatible with GATE version 3.1 and later. GATE 3.0 can be downloaded from the regular GATE download page.

The applications make use of the ADUNA visualisation software package, a trial version of which is available for free and can be downloaded from
http://aduna.biz/products/technology/index.html.

To use the visualisation software presented here, GATE 3.0 must first be installed. Second, the relevant libraries should be placed in the directory gate/lib:

- the jar files from the Aduna lib directory aduna-clustermap-xxx
- the files rio.jar and rio10.jar from the lib directory in the zipped visualisation package containing the visualisation tools.

The software should work under any operating system, though it has currently only been tested on Windows and Linux.

2.3 CorpusAnnotationComparison

The first phase of the ontology learning method involves extracting a set of terms from a corpus. These are then used as a basis for establishing concepts. Naturally the quality of the extracted terms influences the quality of the final ontology. So, when evaluating the ontology learning, this first step should be evaluated as well. This is especially important when comparing the use of different methods to derive these terms.

The FunctCorpusAnnotationComparison utility does the same thing as the Corpus Benchmark tool[CMBT02] available in GATE, but presents the results in a different way. Given two corpora with some terms annotated, it computes the correct, missing and spurious terms, and presents the output graphically using the Cluster Map format.The advantage is that this allows the user to directly access the three main term sets, to visualise recall and precison and to visually compare the performance of different methods.

2.3.1 Using the Comparison Tool

The FunctCorpusAnnotationComparison is a plugin that performs the comparison of annotations. It requires two annotated versions of the same corpus (e.g. the system generated corpus annotations and the gold standard ones, or two different sets of system generated annotations) and compares one version with the other. A sample set of two corpora is provided in the directory PerformanceTest.

To run the tool in GATE, carry out the following steps:

- Install the comparison tool plugin using the GATE plugin manager.
- Create a pipeline (not a corpus pipeline) then load the two chosen corpora as language resources. For example, load the test corpora from the directory ”PerformanceTest” by creating two new corpora and then populating them with the files in ”clean” and ”marked” (respectively) in this directory.

- Load the plugin as a processing resource and add it to the pipeline. With the test corpora provided, the fromCorpus parameter should point to ”marked”, and the toCorpus to ”clean”.
- Run the pipeline.
- The results will be displayed in a new window.

2.3.2 Results display

The results will be displayed in a new GUI where you can explore the ontology. Navigate the left-hand side pane to access each concept. When you click on a concept it will appear on the right hand side and the yellow bullets connected to it will all represent a document from where it was extracted. Clicking on that ”bag” of yellow bullets you will see in the pane below the annotations that exist in the corresponding document and the name of the document. Figures 2.1 and 2.2 show the same results viewed in two slightly different ways. Both figures show the annotations (instances) extracted from three different sources, in terms of how they compared with the gold standard set of annotations. The bullets in blue are the correct annotations, the bullets in yellow are incorrect.

The two figures demonstrate that sometimes it is helpful to look at the results in different ways - there are many other configurations of viewing which are also possible with this software. For example, in Figure 2.1, the instances are compared with the Gold Standard set of instances. For each annotation source (Jena, Sesame, Kaon), we can see how many of the instances were correct (shown in blue) and how many were incorrect (shown in yellow). In Figure 2.2 we also see the instances belonging to the same three annotation sources compared with the Gold Standard. However, in this diagram they are shown in a different way. The circles containing the blue dots in the centre of the figure indicate the correct instances belonging to each of the three sources, while the circles containing the yellow instances around the perimeter of the figure indicate the incorrect instances. This visualisation shows more clearly the comparison between the correct instances of each source, and the comparison between the incorrect instances from each source, while the first visualisation shows more clearly the comparison between correct and incorrect instances from each source.

2.4 OntologyBuilderDisplay

This tool enables the visualisation of the ontology once it has been generated from the corpus. This visualisation supports the evaluation of the ontology by a domain expert by enabling him to see how certain concepts were derived, in that it allows access to the documents where the concept appears. The expert can then analyse how certain concepts

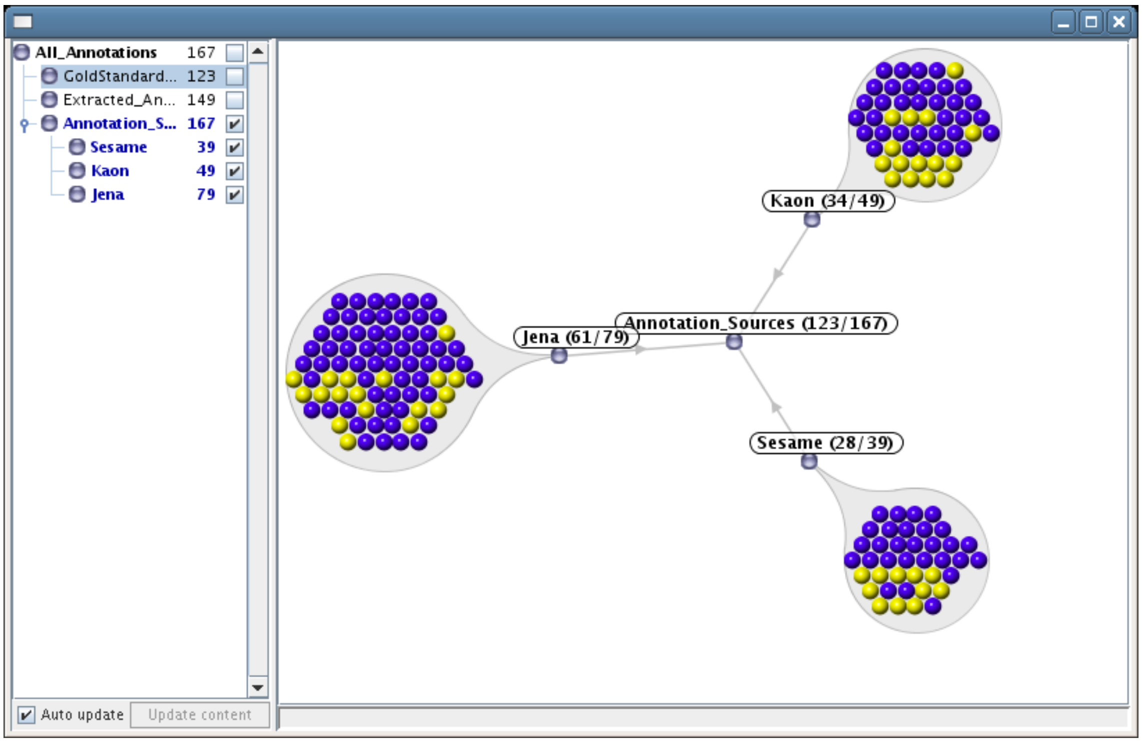

Figure 2.1: View of Corpus Annotation Comparison Results

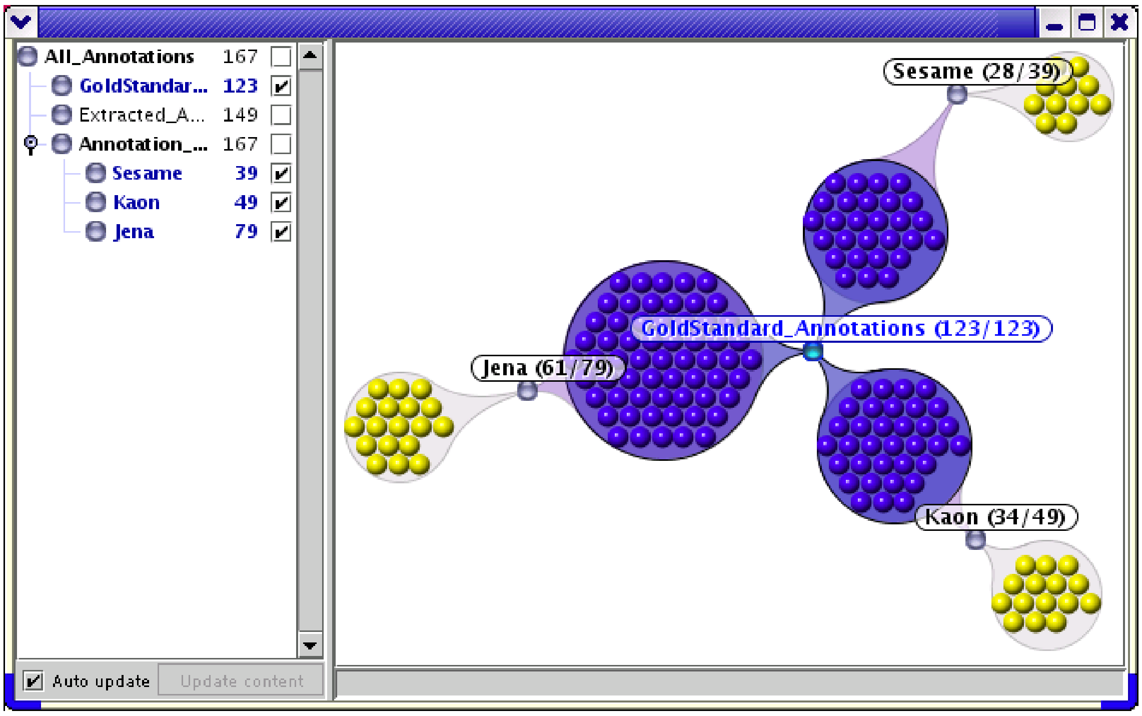

Figure 2.2: Alternative View of Corpus Annotation Comparison Results

interrelate at the document level, which can lead to the derivation of further conceptualizations.

In this situation, the evaluation is done by an expert who relies on his knowledge to decide if a concept is relevant for the domain. Therefore he does not perform a comparison with a Gold Standard ontology. The tool was built to help the expert understand the extracted ontology and to decide which concepts to keep and which ones to delete. Naturally, when deciding on the correctness of the concepts he performs an evaluation, but this is not in the sense of a comparison with a Gold Standard. This kind of evaluation is quite common because it is rare to find pre-existing gold standards against which a comparison can be made, and they are time consuming and expensive to produce. Even if they do exist, they may also be flawed, or there may be more than one possible correct solution.

2.4.1 Using the OntologyBuilderDisplay tool

To run the tool in GATE, carry out the following steps:

- Install the OntologyBuilder Display plugin using the GATE plugin manager.
- Load the corpus provided in the directory ”corpus” as a language resource
- Create a regular pipeline (not a corpus pipeline)
- Load the plugin as a processing resource and add it to the pipeline.
- Configure the runtime parameters as follows:
  - functionalityStyle sets the style of the functionality type concepts (verb only will only use the verbs, e.g. Delete; if you add there any other string then both the verb and its parameter will be used, e.g., DeleteSentence)
  - myCorpus - the name of the corpus
  - nounLexicon - the location of the file lexicon.nouns
  - pruning - sets pruning on and off toggle
  - rdfOntology - the location where the extracted ontology (rdf file) should be saved
  - verbLexicon - the location of the file lexicon.verbs
- Run the pipeline.
- The results will be displayed in a new window.

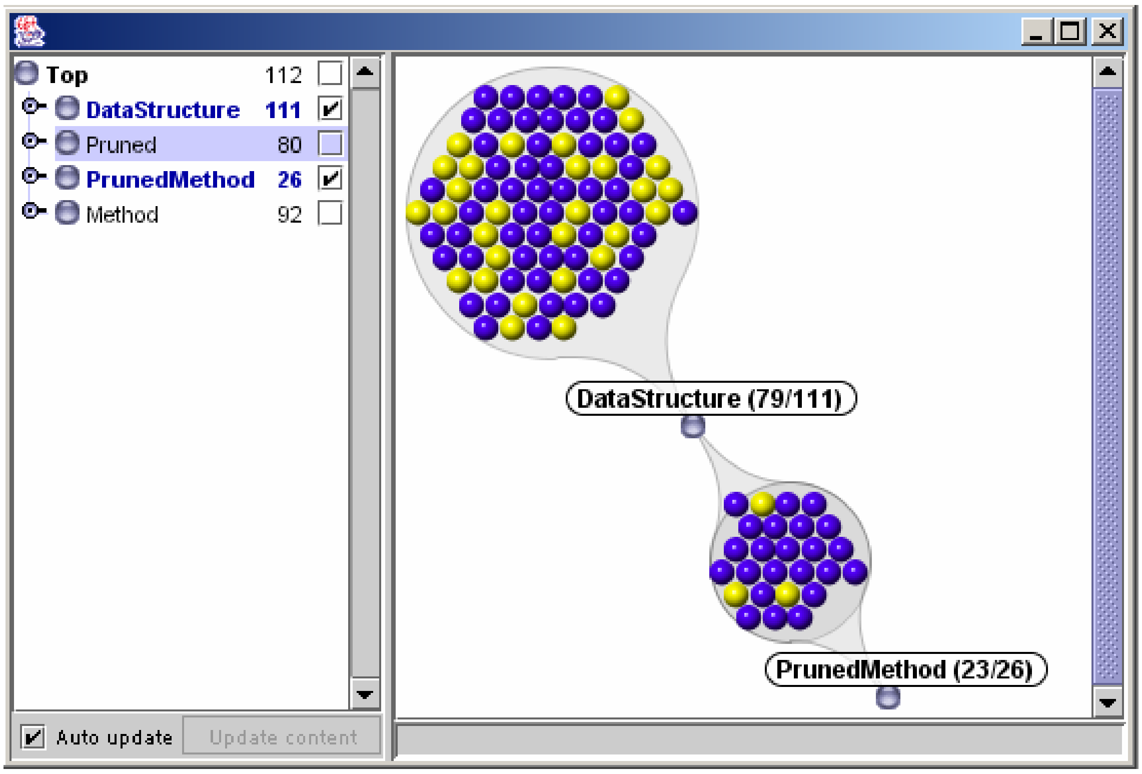

Figure 2.3: View of OntologyBuilderDisplay Results

2.4.2 Results display

The results will be displayed in a new GUI where you can explore the ontology. Navigate the left-hand side pane to access each concept. When you click on a concept it will appear on the right hand side and the yellow bullets connected to it will all represent a subconcept pertaining to that concept. Clicking on that ”bag” of yellow bullets you will see in the pane below the annotations that exist in the corresponding document and the name of the document. Figure 2.3 shows a screenshot of the results. In this figure we can see the relationship between each of the concepts selected (DataStructure and PrunedMethod) with respect to the concept Pruned (which is highlighted in the left hand pane. The blue and yellow bullets this time correspond not to correctness or incorrectness with respect to a gold standard, but instead to the relationship with concepts from Pruned. All the blue bullets indicate concepts which are shared between Pruned and DataStructure or PrunedMethod respectively, while the yellow bullets indicate those concepts which only belong to DataStructure or PrunedMethod. We can see clearly from the figure that although PrunedMethod contains many fewer concepts than DataStructure, a higher percentage of its concepts also belong to Pruned.

2.5 OntologyBuilderSourceDisplay

In some cases, ontologies will be derived from a combination of different document sources. Naturally, concepts that are present in all (or most) sources should be the most significant for ontology building, because an ontology represents a ”shared conceptualization”. The OntologyBuilderSourceDisplay utility shows the overlap of concepts extracted from different sources, allowing the user to filter out those appearing in most sources.

2.5.1 Using the OntologyBuilderSourceDisplay tool

To run the tool in GATE, carry out the following steps:

- Install the OntologyBuilderSourceDisplay plugin using the GATE plugin manager.
- Load the corpus as a language resource
- Create a regular pipeline (not a corpus pipeline)
- Load the plugin as a processing resource and add it to the pipeline.
- Configure the runtime parameters as follows:
  - functionalityStyle sets the style of the functionality type concepts (verb only will only use the verbs, e.g. Delete; if you add there any other string then both the verb and its parameter will be used, e.g., DeleteSentence)
  - myCorpus - the name of the corpus
  - nounLexicon - the location of the file lexicon.nouns
  - pruning - sets pruning on and off toggle
  - rdfOntology - the location where the extracted ontology (rdf file) should be saved
  - verbLexicon - the location of the file lexicon.verbs
- Run the pipeline.
- The results will be displayed in a new window.

2.5.2 Results display

The results will be displayed in a new GUI where you can explore the ontology, The format is the same as for the OntologyBuilderDisplayTool, the difference being that the user can see how the concepts from different sources are related (i.e. which ones overlap).

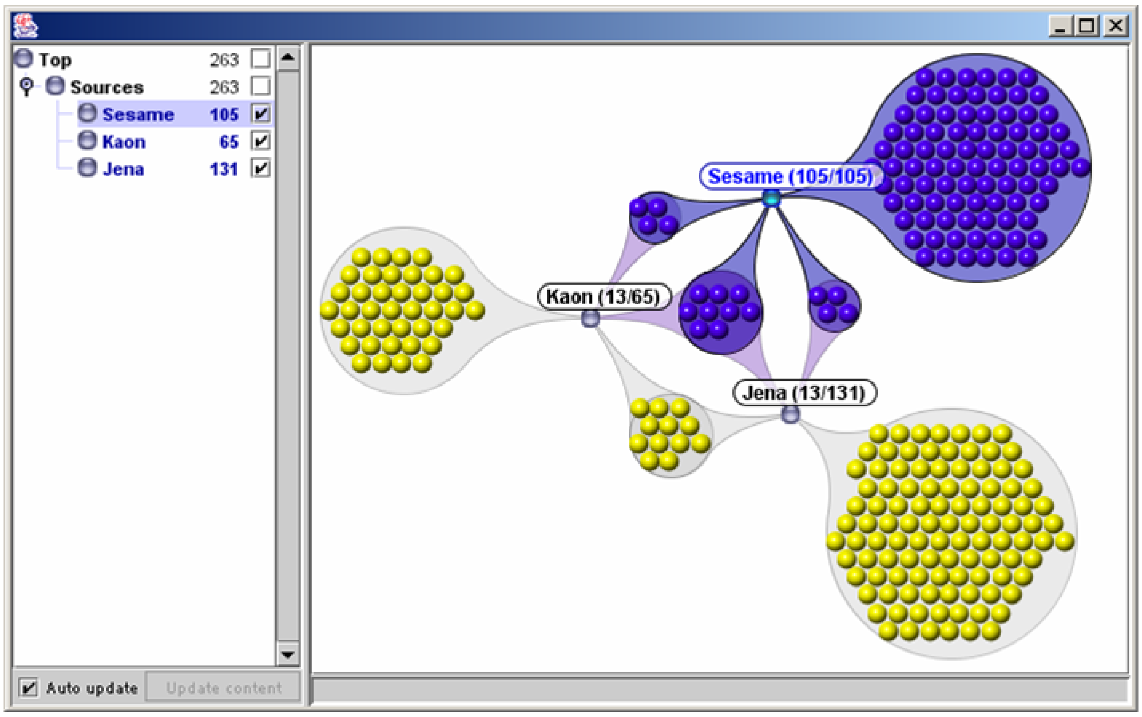

Figure 2.4: View of OntologyBuilderSourceDisplay Results

Figure 2.4 shows a screenshot of the results. In this figure, we see the concepts arranged according to their source. There are three source types: Kaon, Sesame and Jena. We can visualise here the interrelation of the three source types with respect to Sesame (highlighted in the left hand pane). The blue bullets represent those concepts contained in the Sesame sources, So for example the three small circles in the centre containing blue bullets show (from left to right) the set of concepts which appear in both Kaon and Sesame (but not Jena), the set of concepts which appear in Kaon, Sesame and Jena, and the set of concepts which appear in Sesame and Jena (but not Kaon). If we were to highlight the Jena concept in the left hand pane, we could visualise the interrelations with respect to Sesame, and so on.

2.6 Conclusion

These tools are a first prototype of visualisation software used to aid the user in evaluation of ontology building tools, either by enabling the expert to compare the terms extracted, from which the ontology will later be built, or (in the second stage of the ontology building process) to examine the built ontology. Some more detailed descriptions of the software, and a discussion of how such visualisation tools can aid the evaluation process, can be found in [Sab05]. Note that currently the tools are not quite generic, in that some parts are hardcoded, for example to deal with specific annotation types. Future work will focus on making the tools more generic so that they are more widely applicable.

Chapter 3

Evaluation tools

3.1 Introduction

This chapter describes a set of evaluation tools designed to measure how well an ontology has been generated or populated, compared with some kind of gold standard. The tools are currently prototypes only and are not yet completely generic, so that they may require ontologies and other input data in specific formats. Making these tools more generic will be part of future work. The tools are available for download as zipped packages from http://gate.ac.uk/projects/knowledge-web/tools.html

3.2 Comparison of generated ontologies

The ontology comparison tool was developed by researchers at the University of Karlsruhe as a means of testing how well an ontology has been generated with respect to a gold standard ontology, or simply to compare two ontologies. It compares the ontologies at the structural level, i.e. in terms of the concepts and their positioning in the hierarchy, and does not take into account the instances with which it is populated (if such exist). The measures used are defined in [MS01].

The program in its current state is a Perl program which compares two ontologies by getting as input two ontologies in a simple is-a format. Example ontologies are provided with the software. Future work involves integrating the software into GATE as part of the evaluation toolkit. There are two sample ontology files available for download (in onto_compare.zip). Note that the ontologies must be strict taxonomies and a concept may not appear as a subclass of more than one concept.

3.2.1 Running the tool

To run the program, use the following command:

```
perl compare_onto_maedche.perl ontology1 ontology2 > result
```

where ontology1 is the generated ontology, ontology2 is the gold standard ontology, and result is the name of the file where the result will be stored. The format of the ontologies should be as the following:

```
is_a(car,vehicle)
is_a(lorry,vehicle)
is_a(vehicle, mode_of_transport)
```

The results will be written to the result file, and will contain a list of the concepts analysed, the number of concepts in each ontology, the semantic cotopy overlap, the standard cotopy overlap, clustered Precision and Recall, the LTO, TO (taxonomic overlap) and Average TO (semantic similarity). Precise explanations of these metrics are given in [MS01].

3.3 Learning Accuracy evaluation software

The Learning Accuracy (LA) tool is a Java implementation developed by researchers at the University of Karlsruhe, which calculates Learning Accuracy [HS98] for one or more populated ontologies, compared with a gold standard. The Learning Accuracy metric is described in detail in [CLS05]: we present a brief description here.

3.3.1 Learning Accuracy Metric

Cimiano et al [CST03] use Learning Accuracy to evaluate how well an ontology has been populated. It was originally used by Hahn et al [HS98] to measure how well a concept had been added in the right level of the ontology, but it can be equally applied to measure how well the instance has been added in the right place. Learning Accuracy (LA) essentially measures ”the degree to which the system correctly predicts the concept class which subsumes the target concept to be learned”.

LA uses the following measurements:

- SP = the shortest length from root to the key concept
- FP = shortest length from root to the predicted concept. If the predicted concept is correct, then FP = 0, i.e. FP is only considered in the case that the answer given by the system is wrong.

- CP = shortest length from root to the MSCA (Most Specific Common Abstraction, i.e.the lowest concept common to SP and FP paths)
- DP = shortest length from MSCA to predicted concept

If the predicted concept is correct, i.e. if FP =0,

$$LA = \frac{CP}{SP} = 1 \tag{3.1}$$

If the predicted concept is incorrect,

$$LA = \frac{CP}{FP + DP} \tag{3.2}$$

Essentially, this measure provides a score somewhere between 0 and 1 for any concepts identified in an incorrect position in the ontology. If a concept is missing or spurious, the score is 0, and if it is correct, the score is 1 (as with Precision and Recall). So LA provides an indication of how serious the error is, and weights it accordingly.

In its current implementation , an instance of the LearningAccuracy class should be created by passing the ontology as a HashMap, as well as the concepts comprising the ontology. Then the method getLearningAccuracy should be invoked between two Strings representing concepts. The tool has been implemented by the University of Sheffield, and will later be incorporated as part of GATE.

3.3.2 Running the tool

To compile the tool, type:

```
javac LearningAccuracy.java CalculateLearningAccuracy.java
```

To run the tool, type:

```
java CalculateLearningAccuracy <ontology> <correct>
<system1> [<system2> ...]
```

where:
<ontology> is the ontology concept hierarchy, in is_a(A,B) format;
<correct> is the gold standard populated ontology in instance(inst,concept) format. The same instance may be assigned to multiple concepts;
<systemN> are the result files to score against the gold standard, in the same instance(I,C) format.

```
Distance: city location=1
Distance: location thing=1
Distance: country location=1
Distance: holiday day=1
Distance: day thing=1
Calculating learning accuracy for ../la_sample_files/test_annotations
Luc: country
Luc: city
Luc: thing
Learning accuracy: 0.7777777777777778
```

Figure 3.1: Sample output of LA tool

The average LA for each system is printed on standard output, along with the distances calculated and the Luc (MSCA). Figure 3.1 gives an example of the output.

There are some sample files available on which to test the program:

- locations2.isa is a mini concept hierarchy (note that there must be a unique root node in the ontology for the program to work);
- test_gs_annotations is a mini gold standard populated ontology;
- test_annotations is a mini result ontology.

3.4 BDM tool

The BDM tool was developed by researchers at the University of Sheffield in order to overcome some of the problems faced by traditional metrics for evaluation when dealing with ontologies. It will be described in more detail in the future deliverable D2.1.6.2 ”Benchmarking of annotation tools”, where we shall discuss different metrics for evaluation of annotation and ontology population. The tool applies a Balanced Distance Metric (BDM) [May05] to compute semantic similarity between two semantic annotations of the same token in a document.

The metric has been designed to replace the traditional ”exact match or fail” metrics with a method which yields a graded correctness score by taking into account the semantic distance in the ontological hierarchy between the compared nodes. These nodes are called Key and Response.

The semantic distance is computed on the basis of the following measurements:

- CP = the shortest length from root to the most specific common parent, i.e. the most specific ontological node subsuming both Key and Response)

- DPK = shortest length from the most specific common parent to the Key concept
- DPR = shortest length from the most specific common parent to the Response concept
- n1: average chain length of all ontological chains containing Key and Response.
- n2: average chain length of all ontological chains containing Key.
- n3: average chain length of all ontological chains containing Response.

The formula is as follows:

$$BDM = \frac{CP/n1}{CP/n1 + DPK/n2 + DPR/n3} \tag{3.3}$$

and will also form part of the future deliverable D2.1.6.2.

The current tool prototype has been implemented in Awk, and takes as input two corpora or documents annotated with concept information from an ontology. Typically these will be a gold standard corpus (Key) and a system annotated corpus (Result), though it could also be a corpus annotated by two different systems or variations of the same system.

The format of the corpora is currently based on the following:

```
start offset\end offset\token\semantic class\file name
```

The attributes start offset, end offset, token and file name are obtainable from the GATE system. The semantic class is derived from the ontology used for the annotation. This ontology should be in a format in which unique chains are represented on a line, starting with the leaf concept, and each concept is separated by means of a backslash.

The included example ontology, "ontology_fragment.txt", contains three chains:

```
1\Airline\Company\CommercialOrganization\Organization\
   Agent\Object\Entity
2\Man\Person\Agent\Object\Entity
3\Abstract\Entity
```

3.4.1 Running the tool

The program runs on any platform supporting Awk.

Command line syntax is as follows:

```
Awk -f compute_BDM.awk dummy ontology annotated_file1
annotated_file2 output_file
```

The program needs the file “dummy” to run. Example files of the other types are included in the release of this tool, and should be run as follows:

```
Awk -f compute_BDM.awk dummy ontology_fragment.txt
key.txt response.txt output_file
```

”output_file” contains the results of this program. For each pair of semantic labels within the same text span, a line with the following 15 fields of information is produced, again with backslash as field separator:

1. type of concept co-occurrence in ontology
2. name of annotated file
3. start offset
4. end offset
5. token
6. Key
7. Response
8. CP
9. DPK
10. DPR
11. n1
12. n2
13. n3
14. most specific subsumer
15. BDM score

The example output looks as follows:

```
NOT IN SAME CHAIN\ft-BT-07-aug-2001.html\2082\2093\
British Gas\Company\Abstract\1\5\1\4.5\7\2\Entity\0.0646651

IN SAME CHAIN\ft-airlines-27-jul-2001.html\19886\19902\
Monarch Airlines\Airline\Company\6\1\0\7\7\7\Company\0.857143

MATCH\gu-swedish-10-aug-2001.html\198306\198319\
Goran Persson\Man\Man\\\\\\\\1
```

3.5 Conclusions and Future Work

This deliverable presents some ongoing work on evaluation and visualisation tools for ontologies and their versions. One of the main aims of this work is to show how evaluation and its visualisation forms an important role in the lifecycle of ontology development and evolution. For HLT (Human Language Technology) applications such as automatic ontology generation and population from text, evaluation is an important stage in the software development process, enabling refinements to be made to the system as appropriate for the application and scenario. In some cases, evaluation can be performed automatically (for example, against a gold standard) and regression testing commonly forms a crucial stage in the development process of the system[MTC+02]. In the case of ontology versioning, the evaluation and visualisation software enables the user to see how two ontologies differ, in terms of both the concept hierarchy and of the instances attached to the concepts, before negotiation takes place (see for example D2.3.7[ELTV06].

Ongoing and future work takes several directions. First, the visualisation software is being adapted and extended to work with the ontologies used in other deliverables in the WP such as the MarcOnt ontology. Second, the evaluation metrics are being improved and extended, and work is currently being carried out to compare the LA and BDM measures with each other and with traditional Precision and Recall measures. Third, the metrics will be properly integrated in GATE, Sheffield's architecture for language engineering which, amongst other things, is used for automatic ontology population and semantic annotation (see e.g. KnowledgeWeb deliverable 2.1.4[GCMW+05]). Finally, CLIE[TPCB06], a tool for Controlled Language Information Extraction as a way of supporting ontology development, is being developed and will be integrated into the ontology lifecycle. This will improve the cycle in several ways: by aiding ontology generation and versioning by enabling existing ontologies to be transformed into a controlled language, rewritten or modified (using the controlled language) and then regenerated back into an ontology. The visualisation and evaluation software described in this deliverable will also be a valuable part of this mini-cycle of ontology development.

Bibliography

[BTMC04] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10(3/4):349—373, 2004. `http://www.gate.ac.uk/sale/jnle-sale/subs/BONTCHEVA--jnle-final.p`

[Chi92] Nancy Chinchor. Muc-4 evaluation metrics. In *Proceedings of the Fourth Message Understanding Conference*, pages 22–29, 1992.

[CLS05] P. Cimiano, G. Ladwig, and S.Staab. Gimme' The Context: Context-driven automatic semantic annotation with C-PANKOW. In *Proceedings of the 14th World Wide Web Conference*, 2005.

[CMBT02] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[CST03] P. Cimiano, S.Staab, and J. Tane. Automatic Acquisition of Taxonomies from Text: FCA meets NLP. In *Proceedings of the ECML/PKDD Workshop on Adaptive Text Extraction and Mining*, pages 10–17, Cavtat-Dubrovnik, Croatia, 2003.

[ELTV06] J. Euzenat, L. Laera, V. Tamma, and A. Viollet. Negotiation/argumentation techniques among agents complying to different ontologies. Technical Report D2.3.7, KnowledgeWeb Deliverable, 2006.

[GCMW$^+$05] R. Garcia-Castro, D. Maynard, H. Wache, D. Foxvog, and R. Gonzalez Cabero. Specification of a methodology, general criteria and benchmark suites for benchmarking ontology tools. Technical Report D2.1.4, KnowledgeWeb Deliverable, 2005.

[HS98] U. Hahn and K. Schnattinger. Towards text knowledge engineering. In *Proc. of 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 524–531, Menlo Park, CA, 1998. MIT Press.

[May05] D. Maynard. Benchmarking ontology-based annotation tools for the semantic web. In *UK e-Science Programme All Hands Meeting (AHM2005) Workshop "Text Mining, e-Research and Grid-enabled Language Technology"*, Nottingham, UK, 2005.

[MS01] A. Maedche and S. Staab. Comparing Ontologies – Similarity Measures and a Comparison Study. Technical Report Internal Report No. 408, AIFB, University of Karlsruhe, 2001.

[MTC^{+}02] D. Maynard, V. Tablan, H. Cunningham, C. Ursu, H. Saggion, K. Bontcheva, and Y. Wilks. Architectural Elements of Language Engineering Robustness. *Journal of Natural Language Engineering – Special Issue on Robust Methods in Analysis of Natural Language Data*, 8(2/3):257–274, 2002. `http://gate.ac.uk/sale/robust/robust.pdf`.

[PSBC03] P.W.Lord, R.D. Stevens, A. Brass, and C.A.Goble. Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation. *Bioinformatics*, 19(10):1275–83, 2003.

[Sab05] M. Sabou. Visual Support for Ontology Learning: an Experience Report. In *Proceedings of the 9th International Conference on Information Visualisation (IV05)*, 2005.

[TPCB06] V. Tablan, T. Polajnar, H. Cunningham, and K. Bontcheva. User-friendly ontology authoring using a controlled language. In *5th Language Resources and Evaluation Conference*, 2006.

[WLT^{+}04] M.M. Wood, S.J. Lydon, V. Tablan, D. Maynard, and H. Cunningham. Populating a Database from Parallel Texts using Ontology-based Information Extraction. In *Proceedings of NLDB 2004*, 2004. `http://gate.ac.uk/sale/nldb2004/NLDB.pdf`.